

# *Advanced document analysis and automatic classification of PDF documents*

**by William Lovegrove, BSc.**

**Submitted to the University of Nottingham for the degree of Doctor of Philosophy, December, 1996**

# Acknowledgements

I would like to thank the following persons for their help and contributions towards this thesis.

My supervisor, Leon Harrison, my research group leader, David Brailsford and my advisor, David Elliman, for expert advice in the domains of electronic publishing and artificial intelligence.

My parents, Gillian and Michael for their constant support and encouragement.

My colleagues in the Electronic Publishing Research Group at the University of Nottingham for all the advice they have given me over the years: Paul, Phil, Steve, Dave, Peter, Wendy and Eddie.

Liz McQuarrie and the Acrobat Engineering Team at Adobe Systems for introducing me to the finer points of the PDF document model.

*William Lovegrove,*  
16/9/96

# Contents

---

|  |           |
|--|-----------|
| <b>CHAPTER 1, INTRODUCTION .....</b>                             | <b>10</b> |
| <b>CHAPTER 2, CONTEMPORARY ELECTRONIC DOCUMENT FORMATS .....</b> | <b>18</b> |
| 2.1 INTRODUCTION .....   | 19        |
| 2.2 ELECTRONIC DOCUMENT DEFINITIONS .....                        | 19        |
| 2.3 LOGICAL DOCUMENTS .....                                      | 21        |
| 2.3.1 SGML .....   | 22        |
| 2.3.1.1 The document type definition .....                       | 23        |
| 2.3.2 HTML .....   | 23        |
| 2.4 GEOMETRIC DOCUMENTS .....                                    | 25        |
| 2.4.1 PAGE DESCRIPTION LANGUAGES .....                           | 25        |
| 2.4.2 POSTSCRIPT .....   | 26        |
| 2.4.3 THE PORTABLE DOCUMENT FORMAT .....                         | 26        |
| 2.4.3.1 The PDF document model .....                             | 28        |
| 2.5 OFFICE DOCUMENT ARCHITECTURE .....                           | 29        |
| <b>CHAPTER 3, DOCUMENT ANALYSIS .....</b>                        | <b>32</b> |
| 3.1 INTRODUCTION .....   | 33        |
| 3.2 LITERATURE SURVEY .....                                      | 33        |
| 3.2.1 DOCUMENT IMAGE PREPROCESSING .....                         | 34        |
| 3.2.2 PAGE SEGMENTATION AND SEGMENT CLASSIFICATION .....         | 35        |
| 3.2.2.1 Top-down segmentation strategies .....                   | 36        |
| 3.2.2.2 Bottom-up segmentation strategies .....                  | 40        |
| 3.2.2.3 Hybrid segmentation strategies .....                     | 43        |
| 3.2.2.4 An assessment of Page Segmentation strategies .....      | 51        |
| 3.2.3 SEGMENTED GEOMETRIC REGION CLASSIFICATION TECHNIQUES ..... | 54        |
| 3.3 INVESTIGATIVE DOCUMENT ANALYSIS RESEARCH .....               | 58        |
| 3.3.1 DIRECTED K-CLUSTERING - RECOG .....                        | 59        |
| 3.3.1.1 Discussion of the RECOG results .....                    | 59        |
| 3.3.2 EXAMINATION OF THE PIXEL PROFILES OF CHARACTERS .....      | 60        |
| 3.3.3 BASIC DOCUMENT ANALYSIS OF PDF .....                       | 64        |
| 3.3.3.1 Typeface comparison .....                                | 64        |
| 3.3.3.2 Prototype segmentation techniques .....                  | 66        |
| <b>CHAPTER 4, DOCUMENT UNDERSTANDING .....</b>                   | <b>72</b> |
| 4.1 INTRODUCTION .....   | 73        |
| 4.2 LITERATURE SURVEY .....                                      | 75        |
| 4.2.1 ISHITANI'S FORM UNDERSTANDING .....                        | 75        |
| 4.2.2 WATANABE'S DOCUMENT DEFINITIONS .....                      | 76        |
| 4.2.2.1 Understanding Form documents .....                       | 77        |
| 4.2.2.2 Understanding Library Cards .....                        | 78        |
| 4.2.2.3 Understanding Japanese newspapers .....                  | 78        |
| 4.2.3 NIYOGI'S NEWSPAPER UNDERSTANDING SYSTEM .....              | 83        |
| 4.2.4 DENGEL'S BUSINESS DOCUMENT PROCESSING .....                | 86        |
| 4.2.5 SEMANTIC NET STRATEGIES .....                              | 87        |

---

## Contents

|  |   |            |
|--|---|------------|
| 4.2.6  | TAYLOR'S DOCUMENT CLASSIFIER .....  | 87         |
| 4.2.7  | ESPOSITO'S LOGICAL RULE BASE .....  | 89         |
| 4.2.8  | SAITOH'S TEXT AREA ORDERING SYSTEM .....                                    | 91         |
| 4.2.8.1  | A discussion of Saitoh's approach .....                                     | 91         |
| 4.2.9  | LAM'S ADAPTIVE READING FRAMEWORK .....                                      | 93         |
| 4.3  | UNDERSTANDING PDF DOCUMENTS: THE PROTOTYPE APPROACH .....                   | 95         |
| 4.3.1  | CLASSIFYING BLOCKS .....  | 96         |
| 4.3.1.1  | Finding main text blocks .....  | 98         |
| 4.3.1.2  | Finding peripheral blocks .....   | 98         |
| 4.3.2  | TAGGING TITLE BLOCKS AND FINDING LOGICAL DEPENDENCIES BETWEEN BLOCKS .....  | 99         |
| 4.3.2.1  | Finding title blocks .....  | 100        |
| 4.3.3  | PROTOTYPE DOCUMENT UNDERSTANDING DISCUSSION .....                           | 106        |
| 4.4  | SUMMARY .....   | 107        |
| <b>CHAPTER 5, FINAL SYSTEM DEVELOPMENT AND ENGINEERING .....</b> |   | <b>109</b> |
| 5.1  | A PRECISE DEFINITION OF THE TASK .....                                      | 110        |
| 5.1.1  | A FORMAL HYPOTHESIS .....   | 112        |
| 5.2  | SYSTEM DETAILS .....  | 114        |
| 5.2.1  | THE STASIS INTERFACE .....  | 115        |
| 5.3  | FINAL SYSTEM DOCUMENT ANALYSIS OF PDF .....                                 | 116        |
| 5.3.1  | INVESTIGATING PDF GRAPHICS .....  | 117        |
| 5.3.2  | PROCESSING TEXT LINES .....   | 118        |
| 5.3.2.1  | 'Democracy units' .....   | 119        |
| 5.3.3  | ANALYSING A BLOCK'S 'LEADING EDGES' .....                                   | 119        |
| 5.3.3.1  | Segmenting a block using 'leading edge' values .....                        | 121        |
| 5.3.4  | ANALYSING A BLOCK'S INTER-LINE SPACING .....                                | 122        |
| 5.3.4.1  | Segmenting a block using inter-line spacing values .....                    | 123        |
| 5.4  | DEVELOPMENT AND DESIGN OF THE BLACKBOARD ARCHITECTURE .....                 | 126        |
| 5.4.1  | BLACKBOARD SYSTEMS .....  | 127        |
| 5.4.1.1  | Knowledge sources .....   | 129        |
| 5.4.1.2  | Blackboard data structures .....  | 129        |
| 5.4.1.3  | The Controller .....  | 130        |
| 5.4.1.4  | Problem solving behaviour and knowledge application .....                   | 130        |
| 5.4.1.5  | Example blackboard systems .....  | 130        |
| 5.4.2  | OBJECT-ORIENTED SYSTEM DESIGN .....   | 132        |
| 5.4.2.1  | Analysis of the knowledge sources .....                                     | 133        |
| 5.4.2.2  | Design of the blackboard .....  | 134        |
| 5.4.2.3  | Design of the knowledge sources .....                                       | 135        |
| 5.4.2.4  | Design of the controller .....  | 137        |
| 5.4.2.5  | Tailoring the blackboard framework to the document processing problem ..... | 138        |
| 5.5  | ADVANCED DOCUMENT ANALYSIS: THE GENERATION OF DOCUMENT FEATURES .....       | 144        |
| 5.5.1  | EXTRACTION OF MEANINGFUL DOCUMENT FEATURES .....                            | 145        |
| 5.5.2  | KNOWLEDGE SOURCE ALGORITHMS .....   | 150        |
| 5.5.2.1  | Block KS .....  | 151        |
| 5.5.2.2  | Text Frequency KS .....   | 152        |
| 5.5.2.3  | Graphic KS .....  | 153        |
| 5.5.2.4  | Structure KS .....  | 153        |
| 5.5.2.5  | Title KS .....  | 158        |
| 5.5.2.6  | Super-title KS .....  | 161        |
| 5.5.2.7  | Image KS .....  | 161        |
| 5.5.2.8  | Caption KS .....  | 162        |
| 5.5.2.9  | Footer KS .....   | 163        |
| 5.5.2.10   | Header KS .....   | 163        |
| 5.5.2.11   | Peripheral KS .....   | 164        |
| 5.5.2.12   | Column KS .....   | 166        |
| 5.5.2.13   | Document Class KS .....   | 166        |
| 5.6  | DEVELOPMENT OF A DOCUMENT CLASSIFIER .....                                  | 167        |



## Contents

|  |  |               |
|--|--|---------------|
| 5.6.1  | SPECIFICATION OF THE TARGET LOGICAL DOCUMENT CLASSES .....   | 168           |
| 5.6.1.1  | Logical newspapers .....                                     | 169           |
| 5.6.1.2  | Logical academic documents .....                             | 169           |
| 5.6.1.3  | Logical brochures .....                                      | 170           |
| 5.6.1.4  | Logical forms .....  | 170           |
| 5.6.2  | CLASSIFICATION TECHNIQUES .....                              | 171           |
| 5.6.2.1  | Basic classifier requirements .....                          | 171           |
| 5.6.2.2  | Production rule expert systems .....                         | 171           |
| 5.6.2.3  | Neural Net Classifiers .....                                 | 174           |
| 5.6.2.4  | Development of the STASIS neural net classifier .....        | 177           |
| 5.7  | SYSTEM OUTPUT .....  | 179           |
| <b>CHAPTER 6, ANALYSIS OF SYSTEM RESULTS .....</b> |  | <b>182</b>    |
| 6.1  | INTRODUCTION .....   | 183           |
| 6.2  | DOCUMENT ANALYSIS RESULTS .....                              | 183           |
| 6.3  | DOCUMENT ANALYSIS ERRORS .....                               | 185           |
| 6.3.1  | BAD API LINES .....  | 185           |
| 6.3.2  | ASSIGNING POOR LINE ATTRIBUTES .....                         | 186           |
| 6.3.3  | BAD GEOMETRIC BLOCK FORMING .....                            | 187           |
| 6.3.4  | DROPPED CAPS .....   | 192           |
| 6.4  | ADVANCED ANALYSIS RESULTS .....                              | 193           |
| 6.5  | ADVANCED ANALYSIS ERRORS .....                               | 194           |
| 6.5.1  | LINKING ERRORS .....   | 194           |
| 6.5.2  | COLUMN FORMING ERRORS .....                                  | 195           |
| 6.5.3  | BLACKBOARD DIAGNOSIS ERRORS .....                            | 196           |
| 6.5.3.1  | Hanging headers .....  | 196           |
| 6.5.3.2  | Captions of diagrams .....                                   | 197           |
| 6.5.3.3  | Peripheral entities .....                                    | 197           |
| 6.6  | NEURAL NET CLASSIFICATION RESULTS .....                      | 199           |
| 6.6.1  | ACADEMIC DOCUMENTS .....                                     | 199           |
| 6.6.2  | NEWSPAPER DOCUMENTS .....                                    | 201           |
| 6.6.3  | BROCHURE DOCUMENTS .....                                     | 202           |
| 6.6.4  | FORM DOCUMENTS .....   | 204           |
| 6.6.5  | SUMMARY .....  | 206           |
| 6.7  | NEURAL NET CLASSIFICATION ERRORS .....                       | 207           |
| 6.7.1  | NEURAL NET SHORTCOMINGS .....                                | 211           |
| 6.8  | A STATISTICAL ANALYSIS OF STASIS .....                       | 212           |
| 6.8.1  | EVALUATING THE STATISTICS .....                              | 213           |
| 6.8.2  | EVALUATING THE RUN-TIME EFFICIENCY OF STASIS' STRATEGY ..... | 216           |
| <b>CHAPTER 7, DISCUSSION AND CONCLUSIONS .....</b> |  | <b>219</b>    |
| 7.1  | RESEARCH SYNOPSIS .....                                      | 220           |
| 7.2  | AN ANALYSIS OF STASIS' DOCUMENT PROCESSING STRATEGY .....    | 222           |
| 7.3  | A UNIVERSAL DOCUMENT PROCESSING SYSTEM .....                 | 226           |
| 7.4  | CLOSING REMARKS AND CONCLUSIONS .....                        | 233           |
| <b>REFERENCES .....</b>                            |  | <b>CCXXXV</b> |
| <b>APPENDICES .....</b>                            |  | <b>CCXLVI</b> |
| APPENDIX I: STASIS SCREEN SHOTS .....              |  | CCXLVII       |
| APPENDIX II: AN EXAMPLE DTD .....                  |  | CCLXVIII      |
| APPENDIX III: RECOG .....                          |  | CCLXX         |

# *List of Tables*

---

## **CHAPTER 1, INTRODUCTION**

## **CHAPTER 2, CONTEMPORARY ELECTRONIC DOCUMENT FORMATS**

|  |    |
|--|----|
| Table 1: Some portable document elements and their geometric attributes..... | 29 |
|--|----|

## **CHAPTER 3, DOCUMENT ANALYSIS**

|  |    |
|--|----|
| Table 2: Pavlidis' rules for merging column blocks.....          | 45 |
| Table 3: Essential properties of the manhattan layout style..... | 45 |
| Table 4: Sivaramakrishnan's zone classes.....                    | 56 |
| Table 5: Useful attributes of fonts.....                         | 61 |

## **CHAPTER 4, DOCUMENT UNDERSTANDING**

|   |     |
|---|-----|
| Table 6: Niyogi's physical newspaper structure.....                           | 84  |
| Table 7: Niyogi's logical newspaper structure.....                            | 84  |
| Table 8: Examples of Niyogi's rules from different knowledge hierarchies..... | 85  |
| Table 9: Examples from Esposito's knowledge languages.....                    | 90  |
| Table 10: A list of the tags used in the prototype.....                       | 97  |
| Table 11: Rules for linking block A to block B.....                           | 101 |

## **CHAPTER 5, FINAL SYSTEM DEVELOPMENT AND ENGINEERING**

|  |     |
|--|-----|
| Table 12: STASIS knowledge sources and the knowledge they 'encapsulate'.....       | 133 |
| Table 13: STASIS blackboard objects.....   | 134 |
| Table 14: STASIS processing stages.....  | 140 |
| Table 15: Document features and their indexes.....                                 | 145 |
| Table 16: The number of documents used to train the STASIS classification net..... | 177 |

## **CHAPTER 6, ANALYSIS OF SYSTEM RESULTS**

|   |     |
|---|-----|
| Table 17: The STASIS report for the STELLA.PDF document.....          | 200 |
| Table 18: The STASIS report for the document SPAIN0.PDF.....          | 202 |
| Table 19: The STASIS report for THOMBRAD.PDF.....                     | 203 |
| Table 20: The STASIS report for TAX11.PDF.....                        | 205 |
| Table 21: STASIS document analysis and classification statistics..... | 213 |
| Table 22: Run-time performances of STASIS.....                        | 217 |

## **CHAPTER 7, DISCUSSION AND CONCLUSIONS**

---

# *List of Figures*

---

## **CHAPTER 1, INTRODUCTION**

|  |    |
|--|----|
| Figure 1: The traditional document processing Venn diagram ..... | 15 |
| Figure 2: A new approach to document processing .....            | 16 |

## **CHAPTER 2, CONTEMPORARY ELECTRONIC DOCUMENT FORMATS**

## **CHAPTER 3, DOCUMENT ANALYSIS**

|  |    |
|--|----|
| Figure 3: The projection profiles of a lower case 'u' in Times-Roman font..... | 62 |
| Figure 4: The projection profiles of a lower case 'u' in Helvetica font. ....  | 63 |
| Figure 5: Typical output from the prototype PDF processing system .....        | 70 |

## **CHAPTER 4, DOCUMENT UNDERSTANDING**

|   |    |
|---|----|
| Figure 6: The influence range of Saitoh's blocks..... | 92 |
|---|----|

## **CHAPTER 5, FINAL SYSTEM DEVELOPMENT AND ENGINEERING**

|  |     |
|--|-----|
| Figure 7: Part of the tree of logical documents .....                                  | 110 |
| Figure 8: The Acrobat Exchange U.I. with the STASIS U.I.....                           | 115 |
| Figure 9: A flow chart of the segmentation routine based on inter-line gap values..... | 124 |
| Figure 10: A prefabricated multi-column page of text segmented by STASIS .....         | 125 |
| Figure 11: A blackboard framework .....  | 129 |
| Figure 12: The blackboard and blackboard object class diagram.....                     | 135 |
| Figure 13: The STASIS knowledge source class diagram .....                             | 137 |
| Figure 14: The controller mechanism .....  | 138 |
| Figure 15: STASIS object diagram .....   | 138 |
| Figure 16: A flow chart of the main text style finding algorithm .....                 | 154 |
| Figure 17: Parts of characters .....   | 156 |
| Figure 18: Various umbrellas of influence .....  | 157 |
| Figure 19: A decomposed page and its column histogram .....                            | 167 |
| Figure 20: A two layer neural network .....  | 175 |
| Figure 21: The Sigmoidal Activation Function .....                                     | 176 |
| Figure 22: The STASIS document data structure .....                                    | 180 |

## **CHAPTER 6, ANALYSIS OF SYSTEM RESULTS**

|   |     |
|---|-----|
| Figure 23: A page portion of 'Le Figaro' showing a single API text line definition error. | 186 |
| Figure 24: A text line with invalid geometric attributes .....                            | 188 |
| Figure 25: Bad block forming example 'A' .....  | 189 |
| Figure 26: Bad block forming example 'B' .....  | 190 |

---

# List of Figures

|   |     |
|---|-----|
| Figure 27: A paragraph formatted with a dropped cap style .....               | 192 |
| Figure 28: A document formatted with 'hanging headers' .....                  | 197 |
| Figure 29: A document portion with a mis-classified header block .....        | 198 |
| Figure 30: A document portion showing the a mis-classified footer block ..... | 199 |
| Figure 31: An academic document, STELLA.PDF .....                             | 200 |
| Figure 32: A newspaper document, SPAIN0.PDF .....                             | 201 |
| Figure 33: A brochure document, THROMBRAD.PDF .....                           | 203 |
| Figure 34: A form document, TAX11.PDF .....                                   | 205 |
| Figure 35: A page of a logical brochure .....                                 | 208 |
| Figure 36: The front page of a logical form document.....                     | 209 |
| Figure 37: The back page of a logical form document .....                     | 210 |
| Figure 38: STASIS statistics charts.....                                      | 214 |

## CHAPTER 7, DISCUSSION AND CONCLUSIONS

|   |     |
|---|-----|
| Figure 39: The first tier of the universal document processing system ..... | 231 |
| Figure 40: The second tier of the universal document processing system..... | 232 |

## APPENDICES

|  |          |
|--|----------|
| Figure A: An SGML mark-up for a figure.....                | cclxviii |
| Figure B: SGML DTD declarations for a figure mark-up ..... | cclxviii |

# Abstract

---

This thesis explores the domain of document analysis and document classification within the PDF document environment. The main focus is the creation of a document classification technique which can identify the logical class of a PDF document and so provide necessary information to document class specific algorithms (such as document understanding techniques).

The thesis describes a page decomposition technique which is tailored to render the information contained in an unstructured PDF file into a set of blocks. The new technique is based on published research but contains many modifications which enable it to competently analyse the internal document model of PDF documents.

A new level of document processing is presented: advanced document analysis. The aim of advanced document analysis is to extract information from the PDF file which can be used to help identify the logical class of that PDF file. A blackboard framework is used in a process of block labelling in which the blocks created from earlier segmentation techniques are classified into one of eight basic categories. The blackboard's knowledge sources are programmed to find recurring patterns amongst the document's blocks and formulate document-specific heuristics which can be used to tag those blocks.

Meaningful document features are found from three information sources: a statistical evaluation of the document's æsthetic components; a logical based evaluation of the labelled document blocks and an appearance based evaluation of the labelled document blocks. The features are used to train and test a neural net classification system which identifies the recurring patterns amongst these features for four basic document classes: newspapers; brochures; forms and academic documents.

In summary this thesis shows that it is possible to classify a PDF document (which is logically unstructured) into a basic logical document class. This has important ramifications for document processing systems which have traditionally relied upon *a priori* knowledge of the logical class of the document they are processing.

---

# Chapter 1, Introduction

Document image processing (DIP) can be defined as electronically managing information which has previously been distributed on paper. This vague description washes over the many component areas within the field of document image processing yet it has served as a metaphorical compass to all researchers, both academic and commercial, who have worked towards the machine comprehension of document images.

Initially, Optical Character Recognition (OCR) was seen as the most important aspect of document processing. Consequently, it is the oldest of the document processing research areas. Contemporary OCR packages produce reliable results when presented with a wide spectrum of character fonts and handwritten text [Clark95, Ellim90]. Early document processing packages extracted the text using OCR algorithms and then stored the text with the original image. A user could then search the text files for keywords and call up the appropriate document image if the search returned a match. This was a primitive attempt to present the user with the æsthetic feel of the document plus the power of computerised search.

This partial solution is weak for a variety of reasons. From an æsthetic viewpoint a bitmap is a poor document storage medium. Bitmaps are large, resolution dependent files. From a practical viewpoint, the user cannot perform complex text searches which take advantage of a document's structure, for example, a librarian may want to view all documents which have been written by a certain author. With a structured document all

## **Introduction**

---

the computer need do is search the author fields in the document's structure for a match with the author's name. A crucial aspect of document processing is the re-creation of the original document's structure. Without structure, the text extracted from a document image by a simple OCR package will not always follow the natural reading flow of the original document, particularly if the original document is multi-columned. More importantly, the computer usability of a document's unstructured text is extremely limited.

## **Document structure and appearance**

In a local system, for example a small company, there will be thousands of documents generated in a single year. These documents will range from loosely structured and aesthetically pleasing managing director reports, to highly structured tax return forms. The ideal storage format for these documents should have the capacity to hold both appearance and structural information with equal gravity.

This research presents no such file format, but emphasizes a point that is becoming more and more important as document technology advances: the logical structure of a document is as important as the geometric (or layout) structure of a document. When one thinks of a document one must think of both these aspects. Appearance denotes the feel and presentation of a document, plus hidden or implicit information which is decoded cognitively by our minds to unfold the logical structure. The logical structure of a document must be stored to help computers traverse, recall and understand the document without the need to process the documents with complex cognitive based structure recognition programs. Some authoring packages, for example FrameMaker+SGML™, cater for structure and appearance within their documents.

A more pressing issue is the recognition of logical structure from a legacy document. A legacy document is the term used to describe a paper document which can only be converted to an electronic format by a document image processing system. Libraries, the military, large companies and Universities all have legacy documents that require structural recognition processing. Only recently has research been directed to

## **Introduction**

---

identifying the logical structure of legacy documents as well as their textual content, and yet to be brought up to date and stored electronically as efficiently as contemporary documents are, their structure must be recognised and recorded.

Documents that only exist as page description languages (PDLs) traditionally have no concept of structure, since sending a document's structural information to a printer is redundant work [Adobe90, Oakle88]. Recently, technology has advanced to a position where PDLs have been purposely designed for electronic storage, electronic display and electronic dissemination. Consequently, PDLs can be added to the list of sources of legacy documents which require the identification of structure.

### **Human perception of document layout**

Newspapers have extremely complex page layouts, for example, multi-column pages, multi-font text and different sized colour images. The purpose of a page's layout is to transfer implicit information about the document's content to the reader. A good example can be found on the front page of a tabloid newspaper. The use of exaggerated text sizes draws the eye to the primary story, thus implicitly creating a hierarchy of articles. In essence the appearance of the newspaper is helping us to understand its structure.

This transfer of implicit information is not limited to newspapers. Studies of readers' perceptions of journal articles and software manuals by Dillon et al. [Dillo93] have shown that such readers conceptualise documents as possessing a prototypical form of structure that aids location of material. Dillon suggests that this structure can be viewed from three different perspectives.

- Structure can be imposed on what is browsed by the reader. Therefore, the reader builds a structure to gain knowledge from the document.
- Structure is a representation of convention. It occurs in a text form according to the expected rules a writer follows during document production.



## Introduction

---

- Structure is the conveyor of context. There is a naturally occurring structure to any subject matter that holds together the raw data of that domain. The context is conveyed so the reader grasps the organisation of the text.

These concepts apply with varying degrees of relevance to different document classes. The notion of structure as convention seems to be perceived by readers of journal articles. The notion of structure supporting contextual inference seems pertinent to users of software manuals. Research in the domain of linguistics and discourse comprehension lends strong support to the concept of structure as a basic component in the reader's mental representation of a text.

Van Dijk and Kintsch [vanDi80], linguists, suggest that readers acquire schemata, or superstructures, through experience. The schemata facilitate the comprehension of material by allowing readers to predict the likely ordering and grouping of constituent elements of a body of text.

*"a superstructure is the schematic form that organises the global meaning of text. We assume that such a superstructure consists of functional categories...[and]...rules that specify which category may follow or combine with what other categories"* Van Dijk [vanDi80]

On the other hand Johnson-Laird [Johns83], a psychologist, proposes what he terms mental models as a further level of representation that facilitates document understanding. The mental models are based on the perception of structure by the reader. They offer a possible explanation to human document understanding from a psychological perspective.

It is not clear exactly how humans perceive and utilise a document's structure. However, it is widely acknowledged that humans have excellent pattern matching abilities. Humans can tell (to a certain degree) the type of a document they are looking at without semantically processing the content of the document. The only information used to make the classification is the layout of the document, coupled with experience of previous examples of the same document class.

## Introduction

---

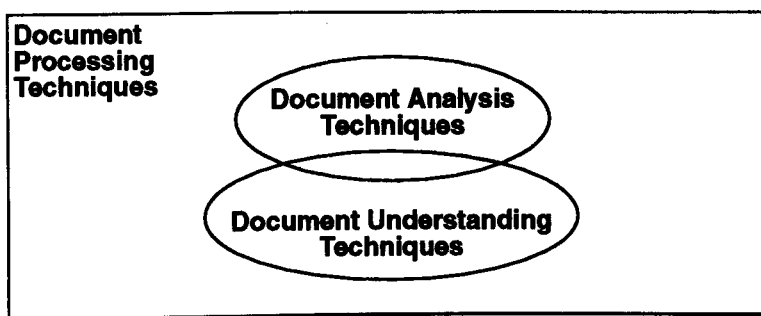
The field of document understanding encompasses the field of structure recognition. By representing structure in a file format the author is dissecting the document and stating the relationships between the logical component parts for the sake of the computer. As humans we do not need to look at the file format to know these component parts, or their relationships. We can recognise the structural relationships in a document thanks to the layout and appearance of the document and our own previous experiences and cognitive abilities.

## Document processing

The task of automated document processing can be divided into two fundamental parts: document analysis (the segmentation of a document image) and document understanding (the logical structuring of the segmented image). This thesis includes two chapters which research these parts ("Thesis structure" on page 16 provides more information on these chapters). Document processing also demands a wide variety of other problem solving techniques, for example, diagram analysis, technical drawing recognition, table recognition, and optical character recognition. This research does not discuss any of these topics but they can be thought of as existing inside the realms of document processing.

Figure 1 is a Venn diagram showing the relationships between document analysis and document understanding for the majority of contemporary document processing techniques. There are elements of both analysis and understanding which are independent of each other. There are also elements which are both analysis and understanding and which lie in a 'grey' area between them, for example, OCR. OCR systems can utilise segmentation techniques (document analysis) and contextual analysis techniques (typically dictionary look-up) to help locate logical words [Ellim90].

The majority of document understanding systems have one thing in common. They all assume *a priori* knowledge of the logical class of document they are processing and are designed to only process examples from that one class of document. Lam [Lam95] describes these document understanding systems as being "closed" systems due to their

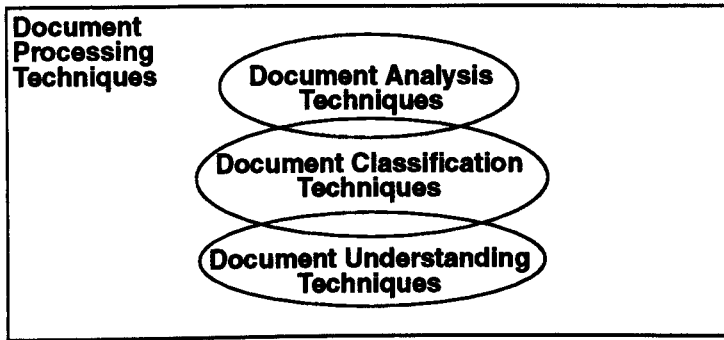


*Figure 1: The traditional document processing Venn diagram*

inability to process documents from more than one class. Engineering a document processing system to logically understand a document from a single class of documents is an extremely complex problem because of the variance that can exist between one document and another within the same class, for example, two different newspapers may have different layouts but they are still newspapers.

This research has concentrated upon finding a method of identifying a document's logical class using only the appearance of the document as a starting point. The practical use of this method extends to helping to create a universal document image processing system which can construct the logical structure of a document from an unclassified document image. The method presented extracts document features from the image using a mixture of established document processing techniques and new algorithms. The features are then used to classify the document. After classification one can apply tried and tested document understanding techniques which are engineered for one particular class of documents.

Figure 2 shows the correct place in the document processing Venn diagram for the proposed classification technique. Aspects of document analysis and document understanding are drawn upon in order to help extract meaningful features. The document understanding techniques used by the system are very basic. They do not require document class specific knowledge and consequently they can be used on all classes of document.



*Figure 2: A new approach to document processing*

### Exploring PDF

Traditional document processing has always started with a bitmap image of the document. Adobe™ Systems Inc. have developed an electronic file format which presents documents to the user in exactly the same format as that in which they were created by the author: the portable document format (PDF) [Adobe93]. However, PDF contains no logical structural information about the document, other than defining logical words. Theoretically, PDF can be thought of as a bitmap. Adobe export an application programme interface (API) with PDF which can be utilised under a licensed agreement with Adobe. The API allows programmers to access the basic components of a PDF document in much the same way as one can access the components of a segmented and decomposed document bitmap image. Whilst achieving the major goal of document classification, this research also documents the experiences of using PDF as the starting point of a document processing system.

### Thesis structure

In Chapter two the spectrum of electronic document formats is described. The definition of a purely logical document is introduced by examining the ISO Standard Generalised Markup Language [ISO86]. Various document formats, which become less structured and increasingly geometrically oriented, are discussed. PDF is described, together with PostScript™, as a document format which contains no capacity for logical structure but which can give a document's author perfect control over the presentation of his or her document. The chapter ends with a review of the Office Document Architecture which attempts to combine logical structure and page appearance in one format.

## Introduction

---

Chapter three defines the concept of document analysis as a sub goal of document processing. A literature review is included which describes the fundamental requirements of a document analysis system as well as discussing contemporary document analysis research. Early research work which contributed to the author's comprehension of document analysis is included in this chapter.

Chapter four reviews contemporary document understanding research. The goals and objectives of document understanding systems are discussed whilst examining the methodologies which other researchers have adopted to reconstruct a document's logical structure. A description of a PDF document processing prototype is included in this chapter. The prototype allowed the author to experiment with a variety of document processing strategies whilst gaining experience from the PDF document model and from approaching the practical problems of document understanding.

Chapter five describes the design and engineering of the final system. The creation of a blackboard framework within the system is documented together with an object-oriented breakdown of the design process. The algorithms used to segment and analyse a PDF document are also explained. This chapter describes the definition of a new level of document processing: advanced document analysis. Advanced document analysis helps extract document features which can be passed to a pre-trained document classifier. The design and development of the document classifier is also described in this chapter.

Chapter six examines the results that the system produces, for a variety of different documents, at each different stage of document processing. Positive and negative document processing results are discussed and classified. Results of the document classification algorithms are included together with a discussion of the strengths and weaknesses of classifying a document based purely on its geometric features.

Finally, the summary restates the major achievements of this research and discusses the practical and theoretical limitations of STASIS's document processing strategy. This chapter also contains the outline of a proposed strategy for universal document image processing in which STASIS's ability to automatically classify documents is vital.

# **Chapter 2, Contemporary Electronic Document Formats**

**This chapter introduces the concepts of logical and geometric electronic documents by examining contemporary examples of both classes of document. In doing so, the goal of many document image processing systems (a purely logical document) is compared and contrasted with the starting state of this research (a purely geometrical document). The ISO Standard 8613 Office Document Architecture (ODA) is discussed and evaluated as a format which is capable of containing the logical structure and geometric structure of a document.**

## 2.1 Introduction

This chapter is not intended to be an exhaustive review of document models, formats and typesetting systems. Instead, this chapter aims to clearly define some basic concepts of electronic publishing which are necessary in order to comprehend this thesis. There are many models, formats and typesetting systems which are not discussed in this chapter but which have played a significant role in the evolution of electronic publishing. They are omitted in order to restrict the content of this chapter to the essential items.

## 2.2 Electronic Document Definitions

*“Knowing the structure of a document is the key to successful computer processing of a document. From different points of view there exist different definitions of document structure” Tang [Tang93]*

Document structure can be realised as two types: geometric (layout) structure in terms of its geometric characteristics (for example, the position and size of each document object), and logical structure due to its logical properties [Tang93].

There are ISO standards for both geometric and logical structures which are taken from the ISO Standard 8613 for ODA [ISO89]. Geometric or layout structure is the result of dividing and subdividing the content of a document into smaller parts on the basis of presentation. A geometric object is an element of the specific layout structure of a document. The following types of layout object are defined:

- a ‘block’ is a basic geometric object corresponding to a rectangular area on the presentation medium containing a portion of the document content;

## Contemporary Electronic Document Formats

---

- a 'frame' is a composite geometric object corresponding to a rectangular area on the presentation medium containing one or more blocks or other frames;
- a 'page' is a basic or composite geometric object corresponding to a rectangular area. It is a composite object, containing one or more frames or one or more blocks;
- a 'page set' is a set of one or more page sets and/or pages;
- the 'document layout root' is the highest level object in the hierarchy of the specific layout structure.

According to ISO standard 8613 (ODA), the logical structure of a document can be defined as

*"...the result of dividing and subdividing the content of a document into increasingly smaller parts on the basis of human perceptible meaning of the content, for example, into chapters, sections, subsections, paragraphs."* [ISO89]

A logical object is an element of the specific logical structure of a document. For a logical object, no classification other than 'basic logical object', 'composite logical object' and 'document logical root' is defined. Logical object categories such as 'article', 'chapter' and 'section' are application dependent.

Most documents such as newspapers, journals, books, and reports are organized hierarchically. Both the geometric and logical structures can be represented as trees. The geometric relationships between blocks can be described by a geometric tree while the logical properties of the document can be represented by its logical tree. Building both the geometric tree and the logical tree is a major task of a document image processing system [Tang93].

The geometric structure and logical structure provide alternate but complementary views of the same document, for example, a document can be regarded as consisting of



chapters containing figures and paragraphs, or alternatively, as consisting of pages that contain text blocks and/or graphics blocks. Correspondence between geometric objects and logical objects may exist, but in general there is no one-to-one correspondence because a logical structure corresponds to a number of geometric structures.

The geometric structure and the logical structure are independent of each other because they have different creation processes. The logical structure of a document is determined by the author and embedded in the document in the editing process. The geometric structure is usually determined by a formatting process. The formatting process may be controlled by attributes associated with the logical structure, for example, each chapter has to start on a new page, or that a section title and the first two lines of its first paragraph are present on the same page [Tang93].

## 2.3 Logical Documents

A logical document contains content elements (text, images, sound, movies) and tags which provide logical markup. According to Goldfarb [Goldf90] these tags have two purposes:

- separating the logical elements of the document;
- specifying the processing functions to be performed on those elements.

SGML (Standard Generalised Markup Language, ISO Standard 8879) [ISO86] will be used as the definitive meta-language for creating a logical document. The markup language HTML (Hypertext Markup Language) will be analysed as an example of SGML's power to create new logical document languages. HTML is a good example to use as it is the 'cornerstone' document format of the World Wide Web. Consequently, HTML has been pressurised and distorted from its original logical definition by users who demand both structure *and* appearance from the documents they use.

### 2.3.1 SGML

SGML is an international standard for the description of marked up electronic text. It is a meta-language that defines the syntax of generalised markup languages [Barro89, Goldf90, ISO86]. HTML is an instance of a markup language. A markup language is a set of markup conventions used together for encoding texts and other document elements. A markup language must specify what markup is allowed, what markup is required, how markup is distinguished from text and what markup means.

SGML can describe logical objects and the structure of a document. It is orientated towards textual data, but provides constructs for identifying the notation of non-textual objects. SGML provides the language to model objects and structure, known as a document type definition (DTD) and the language to identify these objects within a document instance. Logical objects or elements can have additional characteristics associated with them called attributes. Other capabilities exist for handling non-structural portions of a document known as entities, reducing the amount of markup required in a document.

It is worth noting that the document style, semantics and specification language (DSSSL) is designed to specify formatting and other transformations of SGML encoded documents [Roisi93]. For formatting, a DSSSL specification language can create a style sheet language that can be mapped into the DSSSL typographic characteristics and other composition and layout semantics. Additionally, DSSSL [ISO96] includes a language for writing a general transformation specification that allows the user to transform documents from one SGML application into another.

SGML provides the ability to distinguish between the intrinsic content and structure of a document and the specifications for processing that document. With DSSSL, formatting and other processing specifications can be interchanged with SGML documents in a standardised form, while still preserving that essential distinction.

### 2.3.1.1 The document type definition

Within the abstract syntax of SGML there is defined a standard way of specifying the document type definition (DTD). The DTD defines the logical structure of a document in terms of the elements that comprise it (for example, paragraphs, headings, footnotes and so on) and their relationships (for example, the case where a second level heading can only occur within the scope of a first-level heading). It also associates a generic identifier with each element, thus defining the tags that will be used for the descriptive markup of a document [Barro89].

In order to show how a DTD defines the structure of a document an example used by Barron is provided in “Appendix II: An example DTD”. Barron originally took this example from Annexe A of the ISO SGML standard [ISO86].

### 2.3.2 HTML

HTML is a markup language developed at CERN, the European Laboratory for Particle Physics in Switzerland [Graha95]. It allows hypertext links to be followed between documents which can reside anywhere on the internet and are identified by a universal resource locator (URL). It is not the place of this research to investigate or evaluate collective HTML documents together with the hypertext transfer protocol as a hypertext system. What is of more interest is the acceptance and evolution of a structured document language that operates in, and is influenced by, the internet community.

HTML designers have had to serve two audiences: people and computers. The HTML language is geared towards creating machine readable documents rather than aesthetically pleasing documents [Statc96]. There is increasing pressure, coming from users, to evolve HTML from consisting of mainly semantic tags to including capabilities which SGML never set out to accomplish, for example, giving the author the power to define how the document is to be presented to the reader [Sperb94]. HTML document authors have always had the ability to markup bold and italic text and in its latest revision (HTML 3.2), they now

have the ability to set table widths. Good SGML documents should be totally independent of any formatting process.

Web browsers have tried to improve the situation for document designers by offering their own custom tags. Browser companies add more functionality to their browsers with the hope that if their own browser is the only one capable of correctly displaying pages, their product will prevail. This competitiveness continues to alter the definition of HTML.

There are HTML solutions to the problem of appearance in logical documents. W3MAGIC provides a set of HTML tags which enhance web pages and are independent of the browser [W3Mag96]. The software required to interpret the tags is held in a plug-in<sup>1</sup> which is accessible over the internet. When the user downloads a document which contains the special tags, the browser notifies the user that a plug-in is required and asks permission from the user to locate and download the specific plug-in.

From an electronic publishing point of view a more satisfying solution is HTML style sheets. They allow authors to clearly split content and structure from form and appearance. The web browser companies must implement the style sheets into their browsers. DSSSL can be used to formulate these style sheets [ISO96].

The WWW is on the brink of expanding its content base and this may save HTML from corruption. Soon, Adobe Acrobat™, the virtual reality markup language (VRML), Java™, Macromedia Director™ and other data formats will take hold as support for them in popular browsers emerges. There will be less pressure on HTML to be all things to all people [Behle95].

---

1. A "plug-in" is portion of code that can be 'loaded' by a parent application and which increases the functionality of the parent application.

## 2.4 Geometric Documents

A pure geometric document is a document which contains only appearance information, and no logical information. Page description languages (PDLs) [Oakle88] and bitmap formats (JPEG, TIFF, GIF, BMP) [Keyes94] are examples of this class of document. Bitmaps are the traditional input media to document processing systems. Although PDLs, particularly PDF [Adobe90] and PostScript [Adobe93] may contain letters arranged in such a way inside their file format that they form logical words they equally have the capability to be arranged in a totally non-logical formation. Consequently PDLs cannot be thought of as containing any logical information.

### 2.4.1 Page Description Languages

Traditionally page description languages were intended as printer languages. They were designed to facilitate the integration of complex text and graphics for use with laser printers. They can be considered as a communication of an already formatted page or document description from a composition system to an output device, either screen or printer [Oakle88].

PDLs have evolved beyond the printer and are now being used to electronically disseminate documents. PDF [Adobe93] and Digital Paper™ [Commo96] are two examples of PDLs designed for electronic display rather than printing. All forms of PDL have one aspect in common; none contain the capacity to store logical structure. In fact the only geometrical structure they are guaranteed to contain extends to the page level in the geometric structure hierarchy. PDLs are at the other end of the document spectrum from SGML. Oakley gives a good description of the evolution of PDLs into the late eighties [Oakle88].

### 2.4.2 PostScript

PostScript was designed at Xerox<sup>TM</sup> in the late 1970's by John Warnock. PostScript was and is developed and promoted by Warnock's company Adobe Systems Inc. and is today the *de facto* PDL standard. PostScript is a programming language which was built for expressing graphic images. Powerful typesetting features are built into PostScript for sophisticated handling of characters as graphics. PostScript programs are created, transmitted and interpreted in the form of ASCII text which is device independent. The interpreter executes the PostScript program by manipulating a stack of procedure calls which manage other stacks containing operands and dictionaries. Graphics are normally handled as vectors and curves although bitmaps may be specified. As execution proceeds, the interpreter's painting or imaging functions use graphics state variables to calculate where dots should be placed on a page and set the corresponding bits of a page bitmap stored in the printer controller memory. PostScript has a mathematical foundation allowing commands such as scale and rotate. This ensures resolution independent fidelity [Adobe90, Oakle88].

### 2.4.3 The Portable Document Format

The Portable Document Format (PDF) was developed by Adobe Systems Incorporated specifically to aid in the transfer of documents across platforms. PDF is a file format used to represent a document in a manner independent of the application software, hardware, and operating system used to create it [Adobe93, Adobe96].

Based on the PostScript language, PDF allows for device independence and resolution independence. Using Adobe Type Manager<sup>TM</sup> (ATM) and Multiple Master font technology<sup>1</sup>, PDF allows for font substitution across platforms. PDF font substitution does not cause documents to reformat. Substitute fonts created from special serif and

---

1. Multiple Master fonts attempt to duplicate the appearance of the original font used in the document by adjusting certain attributes of their font metrics.

## Contemporary Electronic Document Formats

---

sans serif Multiple Master fonts retain the width and height of the original font. PDF supports standard compression filters to help reduce file size for images, text, and graphics.

A PDF file contains a PDF document and other supporting data. A PDF document contains one or more pages. Each page in the document may contain any combination of text, graphics, and images in a device and resolution independent format. This is the page description (Adobe96).

The Portable Document Format is based on the PostScript language. Although PDF and the PostScript language share the same basic imaging model, there are several important differences between them.

The PostScript language is a complete programming language. To simplify the processing of page descriptions, PDF omits programming constructs. PDF files contain information such as font metrics, to ensure viewing fidelity. PDF files may also contain objects such as hypertext links that are useful only for interactive viewing. PostScript language files do not contain font metrics or hypertext objects. PDF enforces a strictly defined file structure that allows an application to access parts of a document randomly. PostScript language files are linear. Unlike PostScript language files, PDF files cannot be downloaded directly to a PostScript printer for printing.

Adobe has developed the Acrobat™ suite of products to produce and view PDF. PDF can be generated from a printer driver, distilled from a PostScript program via a specialised PostScript interpreter or generated from a bitmap image using Adobe Capture™: a document image processing program. It can be viewed by either Acrobat Reader™ or Acrobat Exchange™. Only the latter program has the capacity to modify the original file, and even then only in terms of deleting and adding entire pages or editing peripheral navigational aids to the document. The navigational aids include electronic bookmarks, hyper-jumping from hotspots to page images and sticker notes which allow the user to comment on the document [Smith93]. Adobe have provided an

application program interface (API) to the Exchange viewer through which third parties can develop their own specialised functionality to Exchange.

PDF, thanks to its PDL ancestry has no capacity for structure. This has its disadvantages. All of the capabilities of SGML are lost, for example logical hypertext, the presence of logical objects and a logical reading order. Furthermore, its critics claim a lack of compatibility with internet search robots (the automatic, behind the scenes cataloguing of HTML documents by internet search engines), a lack of comparable functionality (for example, interactive form technology and Java™ applet technology) and an increased file size over similar HTML documents. PDF's champions claim that the current content expansion of the internet will solve the indexing problem, functionality is being continuously added to Acrobat (the PDF viewer) with each new release and that PDF file sizes are not significantly larger than 'equivalent' HTML files in the average case. Some of PDF's advocates claim to have found cases in which PDF has a significantly reduced file size compared to alternative HTML documents, thanks to the inbuilt ability of PDF to compress text and graphics. There are other contemporary electronic PDLs, notably Digital Paper™, from Common Ground™ Inc. [Commo96]. Digital paper is to Common Ground as PDF is to Acrobat: a platform independent electronic format that reproduces the document as an exact representation of the original.

### 2.4.3.1 The PDF document model

Although PDF is a pure geometric document format, its internal structure does not exactly match that of the geometric structure tree outlined by the ISO ODA standard (ISO89). The similarity between PDF and ISO standard 8613 extends only to the storage of page objects and a document root. The PDF document model does not store frames or blocks in the same sense as the ISO ODA standard defines them. Instead PDF contains streams of information. Typically large images are stored as separate data streams, although PDF does have the capacity to store small images in standard streams containing text and graphics. The streams correspond neither to geometric blocks or frames nor to logical content. They simply contain the 'soup' of PDF operators and text elements which make up the content of the document.



Through the API, Adobe allow the software developer to access logical words. Adobe have developed an in-house algorithm which processes the text and graphic operator ‘soup’ in the streams and reconstructs logical words based on the geometric positions of characters and the legality of the generated words. There is no guarantee that the words created will be logically correct. Through trials conducted during the development of the final system presented in this thesis, various invalid logical words were detected in a variety of different documents (see section 6.3.1, “Bad API lines” for more details). Table 1 lists the most commonly used PDF document model information accessible through the API. Adobe use a naming convention whereby all elements in the PDF document model are prefixed with the letters PD which stands for ‘page description’.

| PD Element | Useful attributes of that PD element              |
|------------|---|
| PDWord     | Bounding box of the word on the page              |
|            | Point size  |
|            | Font name and metrics                             |
|            | Textual content                                   |
| PDImage    | Bounding box of the image on the page             |
| PDGraphic  | Bounding box of the graphic on the page           |
|            | Nature of the graphic, (i.e. line or curve)       |
|            | A set of control points for the graphic operator. |

Table 1: Some portable document elements and their geometric attributes

## 2.5 Office Document Architecture

If, as predicted, browser companies introduce style sheet implementations into their browsers there will be a joining of separate formats for structure and appearance for internet documents. In this way HTML documents will start to resemble ODA documents. ODA was designed as an interchange format for word-processor documents, and is intended for software-to-software communication rather than for direct use by a human user [Barro89].

ISO standard 8613 ODA [ISO89] has the capacity to represent fully both structure and appearance. The definitions for geometric structure and logical structure described earlier in this chapter were taken from this standard. Both dimensions of an ODA document are stored as trees [Nicho84]. The specific logical structure corresponds to the document's logical view. An ODA document may have a generic logical structure, corresponding roughly to the DTD in an SGML document, which indicates for a given logical document object which other logical document objects may appear as its subordinates. The layout view is represented by the specific layout structure, which conforms to a generic layout structure in a similar manner to their logical counterparts.

ODA was designed for transparent stand alone document interchange, yet its acceptance into the pre-internet boom computer user society was severely hampered by the dominance of corporate *de facto* standards such as Microsoft Word™. ODA may see a revival as the internet content base expands.

There is clearly an overlap between SGML and ODA and it has been claimed that SGML subsumes ODA. This is not completely true, since SGML does not have anything to match the layout structure of ODA. The logical structures that can be described in SGML are much more complex than the simple hierarchies of ODA, and it is unlikely that one would want to use the power of SGML in the context for which ODA was intended [Barro89]. Similarly, ODA does not have the descriptive power of PDF or PostScript. By trying to create a format for both structure and appearance ODA makes compromises to the natural abilities of SGML and PDF.

There are other document formats which successfully combine structure and appearance. The most prominent of these is Adobe's Framemaker+SGML™. Framemaker+SGML has all the formatting power of a WYSIWIG document editor coupled with the structural capability of SGML. One needs a Framemaker+SGML application in order to view these documents and the format is not directly internet compatible (for example, one cannot create 'URL-type' hypertext links). In terms of the electronic document spectrum with SGML at one end and PDF at the other, there is a

## **Contemporary Electronic Document Formats**

---

multitude of document formats which lie in between these extremes and which incorporate different degrees of control over document structure and document appearance. TROFF, L<sup>A</sup>T<sub>E</sub>X, Microsoft Word™, Word Perfect™ and PageMaker™ are all examples of electronic document authoring packages which handle the balance between geometric structure and logical structure differently. Proberts [Probe94] provides a good description of various formatting languages and hypertext models in his PhD thesis.

**This chapter has defined and provided examples of the starting point of this research (geometric documents) and the desired goal of document image processing research (logical documents). The next chapter will outline the initial stage in document image processing: document analysis.**

# **Chapter 3, Document Analysis**

**This chapter includes a literary review of contemporary document analysis research. It will continue with a brief synopsis of early research conducted by the author into the decomposition of monochrome TIFF bitmap images. Bitmaps contain no font or typesetting information but with the help of vectorisation software, outlines of black pixel connected components can be isolated and treated as basic geometric components. Inspection and analysis of individual character outlines are also explored.**

### 3.1 Introduction

Document image processing is a relatively young subject and as such there has been no *de jure* or international ISO standards established to provide guidelines and definitions in this area of research. Work by Cambell-Grant [Cambe95], who helped define ODA, is currently addressing this situation. For clarity this thesis will follow the terminology outlined by Tang [Tang91] in his survey paper of document understanding and document analysis systems.

Document processing is divided into two phases: document analysis and document understanding. Document analysis is defined as the extraction of the geometric structure from a document image; document understanding is defined as mapping the geometric structure into the logical structure. Once the logical structure has been captured its meaning can be decoded by artificial intelligence or other techniques. Tang and Suen [Tang95] acknowledge that the boundary definition between document analysis and document understanding is not clear in all cases but their definitions are applicable to this thesis.

Various methods of image segmentation were researched during early stages of the thesis research. The literature survey represents a synopsis of all the algorithms and techniques which were considered by the author; however, only the “Document Spectrum Plot” (described on page 41) was experimented with.

### 3.2 Literature Survey

Analysing the geometric structure of a document takes place one page at a time. Each page is a geometric element which may contain various layouts of text (of varying point sizes and fonts), images and graphics. It is the task of document analysis to decompose that page into its geometric components and extract the geometric structure. There are a range different techniques which accomplish this task.

### 3.2.1 Document image preprocessing

The traditional source of document images (which are supplied to document analysis systems) are provided by electronically scanning pages into a bitmap format. Frequently the pages to be scanned are photocopied so that the tone of the image can be increased and so make fainter text marks bolder. Additionally the photocopied sheets can be fed into an automatic document feeder to the scanner. However, this process can enhance image noise and increase the chance that the document image will be skewed either at the photocopy stage or the scanning stage.

There are several well known image enhancement algorithms which can be applied to the document image to help improve the clarity of the image. Noise detection and removal is often performed before document analysis begins [Gonza92].

Many document analysis systems integrate skew detection algorithms with other aspects of document analysis and so take advantage of the attributes of geometrical objects which have been found previously and which may help in the determination of the skew angle. These systems typically employ bottom-up or hybrid strategies such as white space analysis [Pavli92], white space tiles [Anton95], k-clustering [O’Gor93] and the Hough transform [Hinds90].

Skew has a detrimental effect upon document analysis, particularly if the analysis system is top-down or model-driven. Column recognition algorithms always assume that the columns are perpendicular to the horizontal; similarly, line finding algorithms assume that lines are formatted parallel to the horizontal. In a skewed document image these assumptions would be incorrect.

Several algorithms to detect the skew angle of a digitised image have been published. One of the major factors in detecting the skew angle is looking at the angle of the base line of text to the horizontal. Baird [Baird87] exploits this feature in an algorithm which examines the power spectrum of abstract points taken from clumps of data from the

image which are presumed to be text characters. Due to the algorithm's reliance upon text the efficiency of this algorithm is lower in images which have an increased page percentage devoted to non-textual matter. Ishitani [Ishit93] acknowledges the presence of multi-composite documents and adds a new parameter based on the document image complexity which is obtained from the number of transitions from white to black pixels. Ishitani claims this parameter helps to estimate skew correctly. This global feature is known as 'crossing counts', and can be used as a guide in the classification of segmented areas.

### 3.2.2 Page segmentation and segment classification

Page segmentation is the process by which a geometric document is decomposed into its geometric component elements and those components labelled, typically into one of three classes: image blocks, text blocks and graphic blocks. Graphic blocks differ from image blocks in that graphic blocks consist of graphic operators such as lines, ellipses and rectangles. Image blocks are typically bitmap images, for example, photos.

The goal of page segmentation and classification is to prepare the document to enable the execution of algorithms specifically designed for a particular geometric element, for example, isolating and labelling a text portion enabling OCR algorithms to be directed at it, or, isolating and labelling a graphic block and applying an Engineering Drawing recognition system to try to identify logical entities. Text regions should not really contain more than one text style unless they are of a significantly low percentage of the overall text in that geometric object. This statement increases the complexity of page segmentation and classification algorithms as font recognition is required to discriminate text styles accurately. Yet this discrimination is vital for successful document understanding.

A fuzzy, vague boundary lies between document analysis and document understanding. OCR is definitely part of document understanding but font recognition can be considered document analysis, and typically both are merged into one process. One thing is certain: document understanding takes the output of document analysis (basic geometric page components) and forms high level logical components and the relationships between them.

Most document analysis methods can be described as one of three broad categories: top-down (or model-driven), bottom-up (or data-driven) or hybrid. Top-down algorithms proceed with an expectation of the layout characteristics of the document and are fast and effective for processing documents which always have a specific layout. Bottom-up approaches progressively refine the data by layered grouping operations which can be time consuming, yet it is possible to develop algorithms which are applicable to a variety of documents [Tang91]. The hybrid approach attempts the best of both worlds.

### 3.2.2.1 Top-down segmentation strategies

A top-down page decomposition strategy starts by hypothesising a series of interpretations at a high level and attempts to verify each by searching the tree of implied hypothesis at a lower level of detail finally consulting evidence at the lowest level (characters or pixels). The tree search is typically depth first and fully back tracking [Tang91].

Hu and Ingold [Hu93] describe a very pure top-down strategy. They give as input to their processing system not only the document images but a complete document description. This description contains details of the geometric proportions of layout objects and which page they can be found on. This complete document breakdown is extremely document-specific and would require the creation of an individual document description for every document instance. Such a system is inefficient in terms of processing unknown documents. Therefore, this section will focus on more flexible model-driven strategies which have a greater degree of 'artificial intelligence' in them.



### Projection Profile Cuts

Projection profile cuts is a popular top-down decomposition method. ‘Projection’ refers to the mapping of a two dimensional region of an image into a wave form whose values are the sums of the values of the image points along a particular direction, commonly either horizontal or vertical. A projection profile is obtained by determining the number of black pixels that fall on a particular axis. The profiles represent global features of the document and play an important role in skew normalization, character segmentation and font recognition. The general document composition rule is that every object in the document is contained in a rectangular area. Blank areas are placed between these rectangles. The horizontal document image projection profile will be a wave form whose deep valleys correspond to the blank areas above and below element rectangles. Because a document generally contains several blocks in the horizontal and vertical directions, the projection profile cut should be executed recursively until all blocks have been located [Tang91].

### XYCut

Sylvester and Seth [Sylve95] present a trainable single pass algorithm for column segmentation. In their approach the document image is initially segmented into large layout elements. Sylvester and Seth use the XYCUT based upon horizontal and vertical projection profiles of the image to produce an XY tree representing the column structure of a page of a technical document. These larger frames are then repeatedly decomposed to produce lines in a depth first, back tracking manner. Sylvester and Seth’s system produces poorer segmentation results with any image not of the technical document layout model. Errors such as over segmentation and under segmentation occur. An example of over segmentation is when a word gap is recognised to as a column gap. An example of under segmentation is when a column gap is recognised as a word gap.

### Run Length Smoothing Algorithm

The run length smoothing algorithm (RLSA) was first used to separate text from graphics. Wong et al. [Wong82] extended this research to obtain a bitmap of white and

## Document Analysis

---

black areas representing blocks containing various different types of data. The basic RLSA idea is applied to a binary sequence in which pixels are represented by 0's and black pixels by 1's. The algorithm transforms a binary sequence  $X$  into an output sequence  $Y$  according to the following rules:

- 0's in  $X$  are changed to 1's in  $Y$  if the number of adjacent 0's is less than or equal to a predefined limit  $C$ ;
- 1's in  $X$  are unchanged in  $Y$ .

When applied to pattern arrays, the RLSA has the effect of linking together neighbouring black areas that are separated by less than  $C$  pixels. With an appropriate choice of  $C$ , the linked areas will become regions of a common data type. The degree of linkage depends upon the value of  $C$ , the distribution of white and black in the document, and the resolution. The RLSA is applied row-by-row as well as column-by-column to yield two different sets of results. Different values of  $C$  may be applied in different directions. The two sets of results are then combined with a logical AND operation. Wong found that if smoothing thresholds are chosen correctly the blocks of different content will be smeared into regions with differing features. A shortcoming of this algorithm is the calculation of the constant value,  $C$ . Too great a value will produce blocks which contain regions of differing content within them. Too small a value will produce blocks which are too small.

## Hough transform

The Hough Transform can be used to detect lines at any angle. It consists of mapping points in Cartesian space  $(x,y)$  to sinusoidal curves in  $r\theta$  space by the transformation  $r = x\cos(\theta) + y\sin(\theta)$ . Each time a sinusoidal curve intersects another at a particular value of  $r$  and  $\theta$ , the likelihood increases that a line corresponding to that  $r\theta$  coordinate value is present. An accumulator array is used to count the number of intersections at various  $r$  and  $\theta$ . The cells in the accumulator array with the highest count will correspond to lines in the original image.

## Document Analysis

---

For the best results, the resolution of  $Q$  should be selected such that the pixels comprising the height of a character should be mapped to a single row in the accumulator array. Roughly this means setting  $Q$  to the point size of the text in question.

The Hough transform approach exploits the fact that documents have significant linearity. There exist straight lines in tables and diagrams. Centroids of connected components corresponding to text also line up.

Srihari et al. have shown that the Hough transform is a representation of the projection profiles of the document in every possible orientation [Sriha89]. The analysis of the accumulator array has an added advantage in that it can provide the angle of skew in the document.

### Form Definition Languages

Higashino et al. proposed a top-down document analysis method where the document layout structure knowledge is effectively utilized to parse the two dimensional physical document structure [Higas86]. They devised a knowledge representation called Form Definition Language (FDL), to describe the generic layout structure of a document. The structure can be represented in terms of rectangular regions each of which can be recursively defined in terms of smaller regions. The basic concept of the form definition language is that both the geometric and logical layout structures of a document can be described using these rectangles. These generic descriptions are then matched to the preprocessed input document images. This method is powerful but entirely reliant upon the efficiency of the matching algorithm. This technique sits between the boundaries of document analysis and document understanding.

Yu et al. [Yu93] have extended Higashino's idea with their Document Architecture Language (DAL) approach to document processing. The DAL supports both regular and irregular document blocks and organises the document blocks in terms of the block relations. However, DAL inherits all of the disadvantages as well as the advantages of Higashino's Form Definition Language.

### 3.2.2.2 Bottom-up segmentation strategies

Bottom-up, or data-driven decomposition starts at the lowest level of detail (pixel clumps or characters) and merges groups of basic geometric components with similar characteristics into larger groups. The features of the data are continually processed as the analysis continues, consequently the classification and segmentation processes are usually one and the same in bottom-up document analysis methods. Neighbourhood line density and connected component analysis are the two commonly used bottom-up methods. Neighbourhood line density (NLD) indicates the complexity of characters and graphics. NLD peaks on character areas are higher than peaks on graphic areas. Character sizes can also be predicted from NLD peak values [Tang91].

#### Connected Component Analysis

A connected component is a set of 8-connected black or white pixels. There is an 8-connected path between any two pixels in every component. Different contents of the document tend to have connected components with different properties. Generally, graphics consist of connected components with a large size [Loveg95a, Tang91]. Text consists of smaller, regular components. By analysing these connected components, graphics and text in the document image can be identified, grouped together into a block and separated from each other.

Connected component analysis is a very popular starting point for a variety of bottom-up strategies. The connected components themselves can be detected and stored by a variety of different methods.

Toyoda et al. [Toyod82] uses a four tuple<sup>1</sup> to represent the size and location of the connected components. The content of Japanese newspapers is classified into five regions: text, abstract, article body, picture and figure. During image analysis the four tuples are merged and classified into these regions according to the features of the regions.

---

1. A tuple is a set. A four tuple is a set of four elements.

## Document Analysis

---

A widely used alternative is the creation of skeletal vector outlines which trace the boundary of connected components reducing them to a series of vector loops. Pavlidis proposed an algorithm in 1986 which detects groups of similar length run lines on adjacent scan lines, which overlap [Pavli86]. Work at the University of Nottingham has developed a variation of Pavlidis's method which looks at pixel runs in the vertical direction as well as horizontal run lengths. A good description of these variations is described by Clarke [Clark95]. Similar work based on contours and skeletons of pixel connected components is described by Hori et al. [Hori93].

Drivas and Amin [Drivas95] argue that a bottom-up approach is much better suited to the segmentation of composite documents which contain graphics intertwined with text, since one can distinguish between the two types before the page is reconstructed.

Drivas uses connected components and a grouping process to determine the skew and form the segmentation algorithm. After the connected components have been determined, neighbouring connected components are grouped together if they have similar dimensions. The grouping algorithm takes one connected component at a time and tries to merge it into a group from a set of existing groups.

Sauvola and Pietikainen [Sauvo95] adopt a similar approach but base their segmentation upon feature classification of connected components.

## Document Spectrum Plot

The document spectrum plot, or 'Docstrum' plot, was developed by Lawrence O'Gorman [O'Gor93]. It is based on the nearest neighbour clustering of connected components. The docstrum is a representation of the document page that describes global structural features of the page and can be used for page analysis. The  $k$ -nearest neighbours are found for each page element. Each nearest neighbour pair  $\{i,j\}$  is described by a two tuple  $D_{ij}(d,q)$  of the distance  $d$  and the angle  $q$  between centroids of the two components. A character might make two or three pairings in a word and across word boundaries within the same line, as well as pairings with characters on

## Document Analysis

---

upper and lower lines. The docstrum is the plot of  $D_{ij}(d,q)$  for all nearest neighbours on the page. It is a polar plot with its origin at the centre; radial distance from this is  $d$  and the counter clockwise direction from the horizontal is  $q$ . The docstrum is so termed because of its similarity in appearance to the two dimensional power spectrum and its analogous utility in globally describing an image. Orientation (skew) and text line information can be determined directly from clusters on the docstrum plot.

O’Gorman uses a value of five for  $k$ . Ideally neighbours would be found to the left, right, above and below each component. The extra neighbour is found for redundancy. The disadvantage of picking a larger value of  $k$  is the extra computational time required to compute the neighbours. Other values of  $k$  may be chosen for different purposes, for example, if text lines are ultimately desired then between line pairs are not needed and a value of two or three for  $k$  is sufficient.

A transitive closure is performed on the ‘line’ nearest neighbour pairings to obtain groups on the same text lines. A regression fit is then made to centroids of each group component to find text lines. This fits a strength line to the centroids in each group by minimizing the sum of the squares of errors between the centroids and the line. O’Gorman uses these text lines and the docstrum plot to make a final estimation of the skew of the page.

He groups lines into blocks based upon three properties to determine if two lines are in the same particular group: a test to see if both lines are parallel, perpendicular proximity and overlap. A feature of the docstrum is that spacing parameters are not required from the user. The docstrum automatically determines dominant spacings from peaks on the histograms of nearest neighbours distances and then uses multiples of these for text line and block detection. The docstrum is also independent of page orientation and the line detection is very robust. However, O’Gorman admits that block detection is less robust and that the whole procedure can be computationally expensive on a image full of text.

### 3.2.2.3 Hybrid segmentation strategies

Both top-down and bottom-up document analysis techniques have weaknesses which have caused some researchers to question the strict application of either strategy within document analysis [Pavli92]. In top-down strategies, verification must finally depend on statistical information and so top-down strategies must unreliably descend to the lowest possible level of detail without triggering frequent backtracking which would increase computation time. Top-down techniques are also widely acknowledged to be weak with highly complex geometric images. Bottom-up strategies are forced to make earlier decisions using evidence from the smallest samples, and so they may suffer from a rapid accumulation of mistakes. According to work by O’Gorman [O’Gor93] and Okamoto [Okamo93] the use of bottom-up strategies only is not enough to guarantee the robustness of segmentation.

#### Bounding Box Projections

Ha et al. [Ha95] can extract words, text lines and text blocks by analysing the spatial configuration of bounding boxes of connected components in a given document image. They recognised the reliance of this particular top-down method upon skew detection. The recognition rate degraded dramatically if a skew angle greater than 0.5% was present. This method is not pure pixel projection technology but it applies the same techniques on a greater scale. Consequently, this method is faster than pixel projection but requires connected component detection beforehand. Connected components are basic data blocks but the bounding box projection reveals larger global structures; in this sense this strategy is hybrid.

#### Pattern Classification

Iwane et al. [Iwane93] propose a layout analysis algorithm based on a pattern classification scheme. They combine the segmentation of an image with the classification of the blocks to help geometrically divide up a page. The classifier defines the feature space in terms of low level image processing features such as connected

## Document Analysis

---

components and projection profiles. In this manner the strategy is data-driven. Iwane targets technical journals and gives the classifier a dictionary that holds reference vectors. In this sense the classifier has a high level model of the page to be decomposed. For a different class of document a different dictionary must be substituted. The basic idea behind the approach is that the layout analysis is put into a pattern classification perspective by treating logical layout components as categories of input patterns. An input pattern is then mapped to a vector in the feature space and classified as a certain category.

### White Space Analysis

Pavlidis and Zhou [Pavli92] propose a method that is independent of skew, unlike RLSA and XY recursive cuts. Their method identifies wide white spaces on adjacent scanlines. The goal of white space analysis is to identify column frames which are as large as possible. The skew angle of the page is estimated from these white streams and the blocks are located as regions between the white streams. The isolated blocks are then placed in accordance with the skew angle.

Pavlidis' algorithm requires a small region elimination process to prune away fragments caused by printing defects. A refining process is employed to merge adjoining regions into very narrow blocks such as those produced by isolated text characters. Pavlidis merges column blocks according to the rules set out in Table 2. The blocks that are narrow in the vertical direction usually contain only a fragment of a single text line and must also be refined. Akindele and Belaïd [Akind93] have devised a white space related algorithm. Their method converts the inter-column and inter-paragraph gaps into horizontal and vertical lines and builds an intersection table from the lines. The entries of this table are used to construct simple polygon blocks with the aid of four connected chain code and a direction table.

Another variation of the white space algorithm is given by Antonacopoulos and Ritchings: the representation and classification of complex shaped printed regions using white tiles. White tiles are the representation of the white space in segmented regions [Anton95].



|  |
|--|
| Notation: Block P and Block Q are column blocks, Q is merged into P when the following three conditions are satisfied. [Pavli92] |
| 1. P and Q are very close in the vertical direction  |
| 2. The centre of column block Q is not far away from the central Y-axis of P   |
| 3. The widths of P and Q are approximately the same.   |

Table 2: Pavlidis’ rules for merging column blocks

Ittner et al. [Ittne93] describe a hybrid layout technique that first analyses white space to isolate blocks and then uses projection profiles to find lines. The page must have a *manhattan layout*. A manhattan layout is briefly described in Table 3. This restrictive definition excludes some types of advertising documents, forms and some broadsheet newspaper layouts. Generally speaking, top-down layout analysis strategies require a manhattan style page layout.

|  |
|--|
| 1. Pages contain blocks of text and lines of symbols.  |
| 2. All symbols are printed upright   |
| 3. Non textual graphics do no occur  |
| 4. Text lines are either horizontal or vertical  |
| 5. Manhattan layout possesses a single transformation that describes skew and shear alignment over the entire image. |

Table 3: Essential properties of the manhattan layout style

Ittner et al. use projection profiles (see “Projection Profile Cuts” on page 37) to further segment text blocks which have previously been segmented using their hybrid system [Ittne93]. Normally for an image of height  $i$ , the horizontal projection  $P_i$  represents the number of black pixels at height  $i$ . Ittner et al. differ slightly from many published methods in that they project their components abstracted as rectangular boxes of the same centre and area in order to reduce implementing system dependent details of symbol shape. From the projection profiles of segmented page zones, the dominant line spacing  $D$  is estimated from the derivative of  $P$ , which is compounded by taking the square roots.

*“D is used as a heuristic to segment blocks into lines motivated by the diversity of text profiles encountered.” Ittner [Ittne93]*

## Document Analysis

---

A smoothed projection is then convolved from  $P$  using a Gaussian kernel with a standard error taken from the dominant line spacing,  $D$ . With this profile and  $D$  the original block is partitioned horizontally by assigning each component to the text line region in which the majority of its area lies.

Ittner et al. also attempt to segment lines into words but only after symbol recognition within the line zones. The algorithm then asks the user for confirmation that words are delimited by word spaces. If there are no word spaces then no segmentation is attempted at all. If word spaces are present a scaleable word threshold is inferred from each text block separately. The threshold must be independent of text size and the text size is estimated from the symbol or font set being used by the document. Consequently, symbol recognition must be utilised before word recognition. However, in attempting to automate language free layout analysis, Ittner has simply reduced the problem to symbol dependent layout analysis.

Okamoto [Okamo93] presents a hybrid algorithm that resembles the recursive XY cut but analyses white spaces as a whole. Okamoto noticed that columns are either separated by white spaces or thin black lines. Through horizontal and vertical scanning of the page he detects these lines and separators. The block connected components are merged in the direction of the separators to complete the segmentation. Okamoto intends his method to be independent of the orientation of the text lines, although they must be either horizontal or vertical. His algorithm shows that a very simple global analysis of a page can produce a good segmentation result.

## Model matching

Although model based analysis immediately suggests that the strategy should be top-down or model-driven by definition, the model is stored separately and used in conjunction with bottom-up techniques to provide evidence to help with the choice of model. The recognition of a document is realised by an analysis system with the help of a model. A model that contains general information about a group of documents is a generic model. Liu-Gong et al. claim that with only a few generic models an analysis

system (which uses a model recognition system) can analyse and recognise many different types of document [Liu95].

Liu-Gong's generic model definition (which should not be confused with the ODA's generic layout structure) is a tree structure containing the characteristics of the layout objects and indicates whether or not a layout object must be present in specific layout structures. Thus while analysing a document the analysis system travels through the hierarchical tree of a model and at the same time it uses the attributes and the methods contained in the class-objects to identify the specific layout objects. Liu-Gong employs histogram analysis and Hough transform skew correction techniques to achieve segmentation. His model matching techniques effectively breaks down any theoretical barriers between document analysis and document understanding as Liu-Gong attempts to logically label his component elements using his generic model.

Farrow et al. have researched another model based hybrid system [Farro95]. Feature tokens are generated from the original image using bottom-up processes. Processing proceeds in a bottom-up manner via a Forward Production System in which production rules utilise the low-level evidence available and detect designated object types in the original image. These objects types are designated high level feature tokens. Farrow uses a matching method to associate the tokens with nodes in his model. In essence Farrow's objective is to fill a defined column area with tokens that have been previously detected: he calls this a column tiling process.

### **A global-to-local approach to layout analysis**

Lam has developed a 'local to global' approach to complex document layout analysis which addresses some of the problems that have troubled other white space analysis segmentation algorithms:

- documents have complex layouts, for example, the white gaps between regions are usually small and some of the regions are non-rectangular;

## Document Analysis

---

- documents are degraded, for example, the detection of white space regions becomes difficult when the white gaps are corrupted by noise.

Lam uses no prior layout knowledge of the document being processed. This is a deliberate strategy on behalf of Lam so that he may use this segmentation technique within an “open” document understanding system. An “open” system is not restricted to processing only one class of document [Lam94a, Lam94b].

Lam’s approach utilises the strength of both top-down and bottom-up strategies and tries to compensate each strategy’s weakness with the other’s strength. The strength of his approach lies in its ability to locate and combine white gap candidates into white space regions. Top-down analysis first divides the page into four equal sized sub-images. Each sub-image is then divided again into four smaller sub images. The partitioning stops when the sub-images are smaller than a predefined threshold. White space zones are identified in the sub-images by looking at the vertical and horizontal pixel profiles of the sub-image. Pixel profile analysis is a top-down strategy. Lam states that

*“top-down analysis does not provide reliable segmentation on noisy and complex documents at the layout level, it can hypothesize probable [white space] candidates when it only focuses on a small area of the page.”* Lam [Lam94b]

The white space zones are then combined in a pair-wise merging operation. The merging operation is performed horizontally and then vertically in two separate stages. The separate sets of results are then ‘aligned’ to see if rectangular regions can be formed.

### **Hirayama’s hybrid column segmentation technique**

Hirayama [Hiray93] has devised a block segmentation method for a document recognition system. His technique is designed specifically for text-segmentation and acts as an input tool for creating electronic databases of various printed Japanese documents.

## Document Analysis

---

The block segmentation process has four stages: detection of character strings; grouping of character strings by height-distance relation; page segmentation using border lines and block unification.

The first stage uses a form of RLSA (see “Run Length Smoothing Algorithm” on page 37 for more details) to group runs of connected black pixels which are separated by an interval of white pixels. The length of the interval is determined by a predefined threshold. The neighbouring lines of black pixels are merged together to form blocks which are then classified (see section 3.2.3, “Segmented geometric region classification techniques” for more details) as either horizontal lines, vertical lines, picture elements or character strings. Character strings can be thought of as lines of text.

Hirayama observed that the character strings were arranged regularly and exploited this feature in order to merge each text area into groups in stage two.

*“The regularity is in the spacing, where the distances between adjacent character strings in a vertical direction within a text area are almost the same”* Hirayama [Hiray93]

Hirayama calculated a threshold value which he used to guide the string merging process. The threshold value is calculated from the analysis of two histograms: a histogram of the heights of the character strings and a histogram of the distances between baselines of adjacent character strings in a vertical direction. Once the threshold has been calculated, two adjacent string lines are merged into the same group if the distance between them is less than or equal to the threshold.

In stage three, the border lines of columns are detected by linking the edges of the text groups. Again, Hirayama bases his strategy upon a key observation:

*“Text areas consist of mainly columns, therefore, the column structure (in other words the edges of the columns)...can be detected by analysing the edges of text groups.”* Hirayama [Hiray93]

Hirayama does not give any more implementation details for this stage other than this statement:

*“if the left or right edges of two or more text groups are in the same vertical column they are linked to one another.”* Hirayama  
[Hiray93]

He does expand into little more detail when he describes the precautions he has implemented in order to preserve the identity of figures which are formatted within, or in between, columns. Essentially, Hirayama over segments his page of text in order to increase the chance of keeping the page's images intact. This over segmentation generates small text areas which are subsequently 'unified' in the fourth stage of the algorithm.

The 'unification' stage is not documented in any detail by Hirayama other than an outline of an algorithm which tracks left to right across the page and top to bottom down the page and unifies text blocks based on their 'spatial relations'.

The most important point which can be concluded from Hirayama's work is that the method of analysing the height-distance relationships between text strings allows text areas to be segmented without predetermined threshold parameters (with the exception of the predefined RLSA-type threshold which is used to segment text lines). Additionally, once the text lines have been extracted, the algorithm is mainly bottom-up or data-driven and can, therefore, be applied to documents which have diverse layout models. The only stipulations that must be observed are that the document is formatted into text blocks and that those blocks form columns. As Hirayama has used a RLSA-type algorithm (which is model-driven) and based his algorithm upon the presence of blocks and columns, his algorithm has inherently taken on some of the characteristics of a model-driven strategy and must be considered as a hybrid strategy.

### 3.2.2.4 An assessment of Page Segmentation strategies

If the document style is *manhattan* with a simple geometric layout then a top-down strategy would be fast and effective. The top-down approach is successful when the recognition procedure can interpret the document structures logically. In other words the analysis system can recognise document images with the knowledge about layout structures by invoking image processing routines as subroutines [Watan93, Tang91, Sylve95].

If the document image is geometrically rich and diverse then a bottom-up strategy would be advisable. The data-driven approach may be used effectively on documents not designed explicitly on the basis of layout structures. If the document style is unknown then a possible solution would be to use white space analysis techniques which produce the best all round performances as they draw on both strategies simultaneously. Each one of the methods described in this section has advantages and disadvantages. All produce errors in certain contexts. It is reasonable to base the choice of a document analysis strategy on the logical class of the document. This statement can be made with a certain amount of confidence thanks to the consistent *ad hoc* formatting rules which exist for certain classes of document.

Chenevoy and Belaïd [Chene91] engineered their system, 'Graphein: hypothesis management for structure document recognition', to take advantage of these *ad hoc* rules to form layout models. Graphein is a blackboard based system which attempts to identify the structure of a document from a generic model based upon the ISO standard for ODA. The system deals with different hypotheses of structure. The system adopts different segmentation procedures according to the hypothesis extracted from the model. A top-down method is applied when a hypothesis is sure enough. A hybrid method is applied when the model is not directly usable and a bottom-up strategy is engaged when the model is unusable.

## Document Analysis

---

The blackboard has several levels of geometric abstraction ranging from page through to character. The hypothesised or found objects are stored as nodes in the corresponding levels, described with attributes and links. The specific structure of the document is expressed in the links which express the hierarchy. In this way Graphein expresses document understanding whilst segmenting the page. This is an attractive system which integrates both aspects of document processing within itself: document analysis and document understanding. However, the document understanding stage is still model based, and like other model based systems it requires *a priori* information regarding the logical structure of the document,

Kise et al. [Kise93] have developed a system for the incremental acquisition of knowledge about document layouts from example documents. The knowledge comprises symbolic descriptions of general layout structures which aid document analysis. The method is incremental in that knowledge is modified using additional examples to cover previously unseen layouts. Counter examples (generated as errors) are reflected in the knowledge so that the system makes no repetitive errors.

Kise's knowledge base contains data on one family of documents; in his paper he uses title pages of journals as an example family. Kise copes with the problem of multiple layout structures for one logical structure by representing each unique layout structure encountered as a class of layout structures.

*"A class description consists of two sub-descriptions: a structure description and a feature description. The former is a description of the nesting relation among layout objects as a tree, while the latter is a description of geometrical features of each layout object and spatial relation between layout objects."* Kise [Kise93]

His system has three phases which are summarised as follows.

- Phase 1, example layout objects are transformed into an instance description.



## Document Analysis

---

- Phase 2, the instance description is matched to each of the existing class descriptions. If a matching class description is found it is generalised using the instance description. Otherwise the instance description is stored as a new class.
- Phase 3, the example image is analysed using the generalised class description to generate counter examples of layout objects (erroneous hypotheses). Then the class description is specialised to exclude the counter examples so that the analysis system may not generate them.

Kise has attempted to cover the potentially endless number of layout variations that a single logical class of documents may have by modelling each instance that it encounters. The advantage of this system is that it is able to guide document analysis using knowledge already acquired from previous experience. This system is well suited to modelling classes of documents which have little variation in geometric layout, for example journal title pages. The disadvantages of such a system would be the continual learning required of class description after class description for documents which have a wide variation in geometric layout for only one logical structure, for example newspapers. This system integrates document understanding knowledge (logical structural knowledge and logical labelling) back into the document analysis processing in an attempt to minimise error creation in the image segmentation.

Akindele and Belaïd [Akind95] have built upon the theories put forward by Kise and developed a system for constructing the generic model for a document class from document samples belonging to each class. They intend to use these models to guide the document analysis processing and improve performances with their Graphein system [Chene91]. For each image sample, the system constructs the corresponding specific physical structure in the same fashion as Kise. The constructed structures are used either to construct an initial model or to infer the generic model. The construction of the initial model is made by combining and transforming nodes in the trees representing the specific structures. The generic model is generated using a method of inference from tree grammars. Akindele states that the reduction of rules resulting from the inference process helps to eliminate redundancy and repetition in the final model.

### 3.2.3 Segmented geometric region classification techniques

Once a document page image has been geometrically segmented it is necessary to classify the segmented regions. Some segmentation techniques use classification features to help segmentation and so the classification stage cannot be separated from the segmentation stage. These techniques are more often than not bottom-up or data-driven strategies in which close examination of the image's base components is of fundamental importance. Typically top-down and hybrid strategies have to employ classification techniques after segmentation. The factors influencing the choice of classification algorithm are the same as those influencing the choice of a document image segmentation technique: class of document; variety of data classes, and the presentation of the input data's segmented regions.

Top-down strategies require classification techniques to help direct the next level of segmentation. Obviously, there is no "next level" of segmentation for half-tone regions of the original image and so the decomposition algorithm must never attempt to perform further decomposition on these zones. However, once text regions have been identified, the decomposition of these regions into lines can commence. Once the lines have been isolated, they can be segmented into regions containing logical words. The decomposition of these regions leads to the isolation of the connected components which represent the glyphs and symbols of characters. Commonly, this continuation of the top-down strategy is not so straight forward. The identification of word regions from line regions is particularly difficult. Many algorithms will identify connected components from the line regions without attempting word region identification. Subsequent OCR of these regions renders the characters within the line. With this extra information algorithms can adopt more sophisticated word identification algorithms such as n-grams (probabilities assigned to specific letter combinations), syntactic analysis and semantic analysis. An early review of word recognition techniques is given by Elliman and Lancaster [Ellim90].

## Document Analysis

---

If the document analysis system is competent and thorough it should identify regions in which graphic elements are present, for example, lines, rectangles, arcs and circles. The isolation and recognition of these graphic areas is non trivial due to the vast variation in combinations of graphic operators and layout appearances.

Many graphic region classification techniques have been proposed. One of the most useful is Fourier analysis, but this is too slow to be attractive [Pavli92]. Wahl [Wahl82] analyses the geometric properties of connected components. The method uses measurements of border-to-border distance within a connected component. These measurements provide a fairly good estimate for the mean line thickness of line shaped patterns, and consequently it has proved to be very powerful in distinguishing line drawings from text. This method cannot efficiently be applied to half tone regions without complex and time consuming calculations.

Other methods use features (measurements of quantity thought to be useful in distinguishing members of different classes) to classify regions. The ratio of black to white pixels within the image provides some useful information. Line drawings such as diagrams have a much lower ratio of black to white pixels than text, while half tones usually have the highest ratio. Yet because of the variability of the ratio in half tones, this feature cannot be used with confidence except to decide that something *cannot* be text or diagrams.

Pavlidis uses the field of statistical pattern classification to categorise regions of document images that he had previously segmented using white space analysis. His method is reasonably fast and efficient [Pavli92]. Pavlidis looks at the correlation between scanlines at  $y$  and  $y+r$ , and defines them as follows.  $L$  is the number of pixels in a scanline and  $p(y,k)$  is the value of the  $k$ th (binary) pixel of scanline  $y$ . The quantity in brackets equals 1 if the two pixels have the same value and -1 otherwise. Thus if two scanlines disagree over length  $k$  and thus agree over length  $L-k$  then the sum is  $L-2k$  and  $C(r,y)$  equals  $1-2k/L$ . The key observation is that for lines of text and diagrams  $C(r,y)$  is a rapidly decreasing function of  $r$ , at least for small values of  $r$ , and  $C(1,y)$  is quite high. For half-tones  $C(r,y)$  is quite flat and even exhibits periodicity.

Another researcher who used white space related techniques to decompose document images continues to use white space related techniques to classify image zones. Antonacopoulos [Anton95] noticed that segmented areas of interest corresponding to different classes of elements have distinct textual characteristics.

*“Text regions contain significant numbers of white tiles which are distributed evenly inside the region and the white area covered by them is large in proportion to the total area of the region.”*

Antonacopoulos [Anton95]

He found that half-tone regions contained less white space than other regions, that the size of the tiles varied considerably and they were not evenly distributed. Regions containing line art diagrams were characterised by the relatively large amounts of space they contained in the form of wide tiles.

Sivaramakrishnan et al. [Sivar95] classify zones into nine classes. By increasing the number of possible zone classes, Sivaramakrishnan et al. have made the classification process more detailed and refined than it would normally be. The advantages of this increase in detail is the increased information which can be relayed to the document understanding phase of document processing. A simple disadvantage is the increased likelihood of error given the increased number of classes. Sivaramakrishnan’s zone classes are displayed in Table 4.

|                                  |
|----------------------------------|
| <b>Text with font &lt; 18pt.</b> |
| <b>Text with font &gt; 19pt.</b> |
| <b>Math</b>                      |
| <b>Table</b>                     |
| <b>Halftone</b>                  |
| <b>Map/drawing zone</b>          |
| <b>Ruling</b>                    |
| <b>Logo</b>                      |
| <b>Others</b>                    |

**Table 4: Sivaramakrishnan’s zone classes**

Belaïd and Akindele [Belaï93] also used an expanded set of possible block classes, but this section will concentrate on Sivaramakrishnan's block classifying algorithms as they attempt a more detailed block classification system.

A feature vector is created for each zone in which the properties of that zone are recorded. Statistical pattern recognition is used to classify each zone on the basis of its feature vector [Sivar95]. A decision tree classifier is constructed using a training set of feature vectors with true class labels. The features which comprise the feature vector are as follows:

- the number of runs on the foreground and background are calculated by adding up the number of runs along each line in the zone. When calculated in all four directions (horizontal, vertical, left-diagonal and right-diagonal) these add up to eight features in the feature vector;
- the total run length for all runs along the background is calculated and divided by the total number of background runs to give the run length mean of the background. This is repeated for the foreground. When carried out in all four directions this provides another eight features in the feature vector;
- run length variance in all four directions for both foreground and background pixels. This provides another eight features;
- spatial mean in all four directions. This gives another four features;
- spatial variance in all four directions. This gives another four features;
- autocorrelation of the line projection, number of foreground runs, spatial mean and run length mean with the line number provides an index  $r$ . This feature is calculated for all four functions and in all four directions giving a further sixteen features. The process is repeated to create another sixteen features by examining the tangent to the autocorrelation function when  $r$  approaches zero;

- the ratio of black pixels to the total number of pixels in the zone provides a single feature;
- the area of the zone (*height*  $\times$  *width*) provides a single feature;
- the quotient of the zone width and the width of its column is calculated providing a further single feature.

Sixty seven features are calculated in total for every zone vector. The purpose of describing the nature of these features in this thesis is to illustrate their reliance upon pixel information. Although the classification of zones into detailed classes can be viewed as the creation of additional information and in that sense is good, the implementation is anchored to the document processing of bitmap images and cannot be transferred to page description technology.

PDF documents store the type of an element within the document model. Consequently, there is never a need to perform classification algorithms upon elements of PDF documents. The algorithms outlined in this section are only applicable to document analysis algorithms which process bitmap images. They are still included in this thesis as they formed an important part of the early research undertaken by the author. In particular, the creation of a feature vector by Sivaramakrishnan to help classify geometric zones was influential upon the decision to create a feature vector to help classify an entire document. See section 5.5, “Advanced document analysis: the generation of document features” for more details on the document feature vector.

### 3.3 Investigative document analysis research

The following subsections describe the practical research undertaken by the author into document analysis and the detailed analysis of character components. Although this research has little practical bearing on the algorithms used in the final system it provided good document processing experience.

### 3.3.1 Directed K-clustering - RECOG

The RECOG, or block recognition, system was based upon the “Docstrum plot” algorithm by L. O’Gorman (see “Document Spectrum Plot” on page 41). Modifications were made to O’Gorman’s algorithm which rendered good results at identifying paragraphs and lines in particular. The algorithm implementation has a graphical interface which portrays the state of the algorithm during the process of decomposition.

A paper describing in the algorithm was presented at a symposium for document image analysis and multimedia environments [Loveg95a]. An original copy of the paper is included in “Appendix III: RECOG”.

RECOG decomposes a page image into its component blocks based upon a nearest-neighbour clustering algorithm. The blocks which this clustering algorithm produces are classified into text and non text types. The text type blocks are then processed by an adaption of the clustering algorithm presented by O’Gorman which is tailored to finding logical paragraphs, lines and words in the text blocks.

#### 3.3.1.1 Discussion of the RECOG results

RECOG produced good results on a variety of different page layouts including multi-column format. The text decomposition technique is designed to be a preparation technique prior to processing the image with other specialised algorithms: optical character recognition; diagram recognition; table recognition; logical structure realisation. The algorithm implementation effectively filters out and partially classifies areas of the image.

There are two separate stages to the modified “k-clustering” algorithm: block isolation and decomposition of text blocks into lines and words. Processing of text blocks into lines is effective and efficient. However, the three neighbour clustering is computationally heavy. Furthermore, the worst case occurs when a full page image (for example a large photo) is passed to the three neighbour clustering algorithm for page

decomposition. The number of neighbour comparisons is immense given that the vectorisation technique divides the page up into relatively small loops.

A decomposition technique based upon white space layout would nullify the worst case and improve the computational time. However, the “k-clustering” technique is robust and produces sound results.

### 3.3.2 Examination of the pixel profiles of characters

The RECOG system created geometric blocks based solely on the positions of the geometric vector outlines on the page. It became clear that after the segmentation and classification of geometric blocks of vector loops, closer inspection of a block’s component loops was required. A closer inspection would reveal more detailed geometric properties, for example characteristics of the typeface of character vector loops. These detailed attributes could be used to further segment large blocks into blocks containing vector loops of similar attributes. Furthermore, it was hoped that the information that the detailed geometric attributes of text loops provided would enable geometric blocks to be compared with one another in terms of geometric prominence on the page. Table 5 lists geometric attribute features of fonts which can be used in font comparison algorithms. In order to find some of these attributes research was conducted into the analysis of character vector loops. This section documents the methods, results and conclusions of that research.

Analysis of normalised loops was used to achieve independence from point size. The algorithm took the loops that made up a character and scaled them to 100 pixels by 100 pixels. The normalised loops were then plotted on an integer grid and the character body was filled in with a simple flood-fill algorithm. Once the character had been normalised, vertical and horizontal projection profiles were created: a method originally proposed by Zramdini and Ingold [Zramd93].



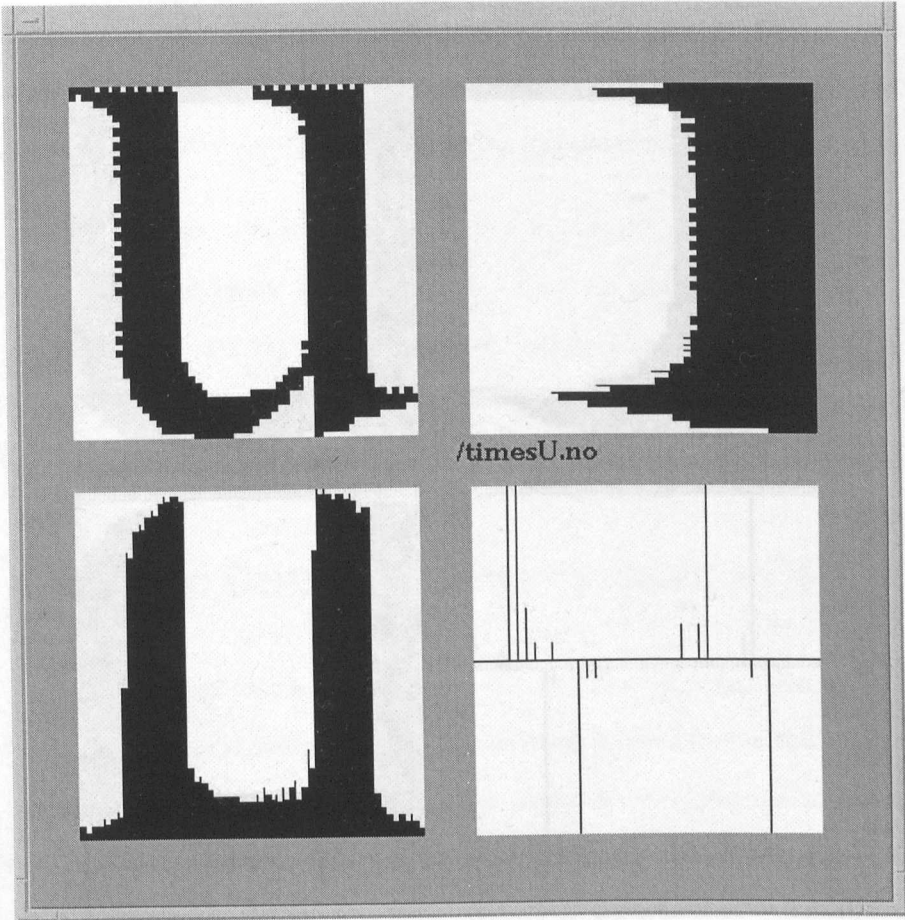
These profiles are essentially bar charts of the pixels in the normalised character projected in either the vertical or horizontal plane. This technique was used by Zramdini and Ingold as they looked at groups of characters on a text line. They discovered that by looking at the first derivative of the horizontal projection they could easily tell whether a character was italic or not. Further research was undertaken to find out (from a character's projection profile) whether or not that character was serif or sans serif.

|  |
|--|
| Type (serif/sans serif)                  |
| x-Height                                 |
| Slant angle of cross strokes.            |
| Contrast between thick and thin strokes. |
| Axis curves                              |
| Serif type, bracketed - non bracketed.   |
| Weight (of stems and serifs)             |
| Character/counter divergence             |
| Stem height and width                    |
| Distinguishing font features             |

Table 5: Useful attributes of fonts

Figure 3, on page 62 was taken from a program that created vertical and horizontal profiles. There are four drawing areas in the figure, the top left is the normalised character (a Times font lower case 'u'), top right is the vertical projection of that character, bottom left is the horizontal projection and bottom right is the 2nd derivative of the horizontal projection profile. The second derivative profile was included to test Zrandini and Ingold's claim that italic characters had a pronounced and different 2nd derivative profile to normal characters.

By examining the horizontal projection profiles of both Figure 3 and Figure 4, on page 63, a profound difference can be seen between serif and sans-serif character profiles. A possible explanation of the difference is that serif characters typically have a slope at the beginning and end of their horizontal projection whereas sans-serif characters do not have this slope; their projections are not gradual. This summation



*Figure 3: The projection profiles of a lower case 'u' in Times-Roman font*

applies only to characters that have pronounced stems on one or both sides of the character. Characters such as 'o' and 'e' have no such stems and thus the projection profiles of these characters in serif and sans-serif fonts do not differ substantially.

By sampling the horizontal projection profile of any letter, an abstract version of that character's profile was created, for example, a normalised horizontal projection 100 pixels in length could be sampled once every 5 pixels resulting in a projection of length 20. This form of sampling gives a richer profile than if the character had been originally normalised to 20 pixels by 20 pixels. Using these sampled profiles, experiments with a fuzzy pattern matcher were performed. A fuzzy logic inference engine was trained on a number of abstracted projection profiles. The inference engine would classify a loop

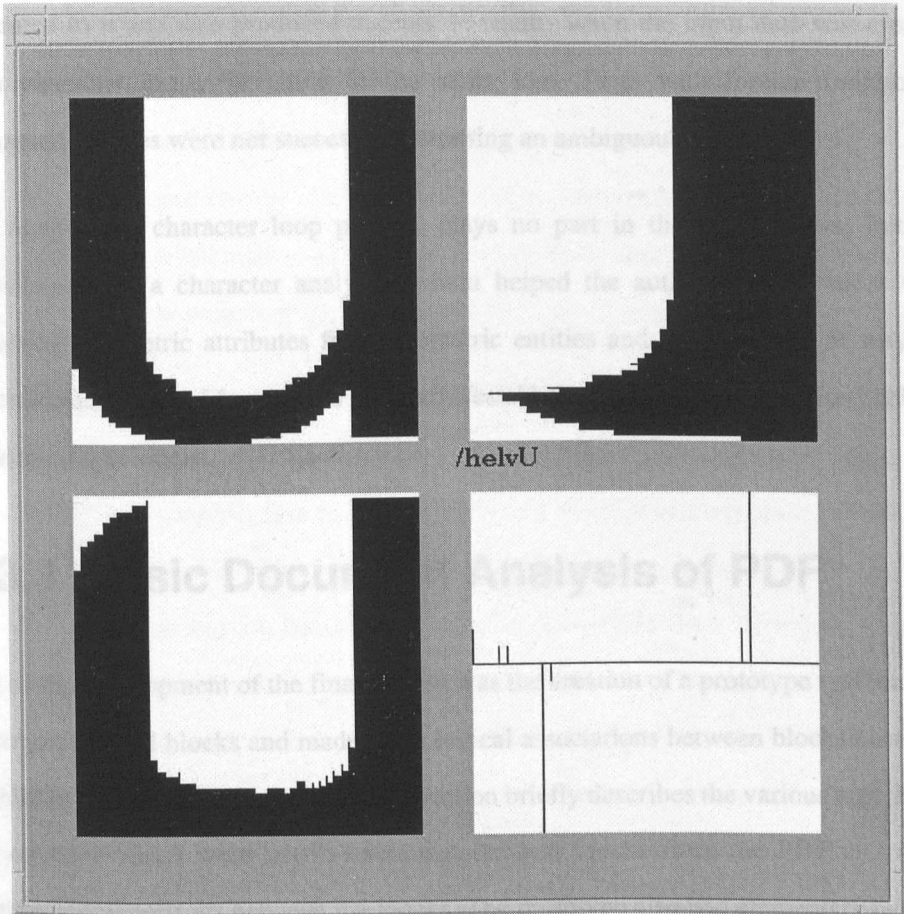


Figure 4: The projection profiles of a lower case 'u' in Helvetica font.

into one of three types: serif, sans serif or ambiguous. Fuzzy logic was proposed for two reasons:

- projections for the same characters at different image resolutions are different. A pattern matcher must be able to deal with patterns for all point sizes at all resolutions for all characters;
- a fuzzy set pattern matcher would be able to accommodate all fonts (regardless of whether they had been seen before) based on a few well chosen abstracted projections. Fuzzy logic theory has shown itself to be robust in dealing with inconsistent and uncertain input data of this type [Klir88, Zadeh83].

The character loop analysis system returned the correct result when the training data was presented to it and also produced acceptable results when the input data was created from character loops formatted in the same font. Tests with foreign (unknown) abstracted profiles were not successful, returning an ambiguous result.

The analysis of character loop profiles plays no part in the final system, but the development of a character analysis system helped the author to experiment with extracting geometric attributes from geometric entities and start to think of ways in which the attributes of fonts could be used to establish geometric relationships between segmented text blocks.

### **3.3.3 Basic Document Analysis of PDF**

Part of the development of the final system was the creation of a prototype system. The prototype formed blocks and made basic logical associations between blocks based on the blocks' geometric information. This section briefly describes the various algorithms and routines which were used to segment the text blocks from the PDF and make geometric comparisons between the blocks. The prototype also had elementary tagging procedures and attempted a basic level of document understanding. A complete description of the early PDF document understanding routines are described in Chapter 4 (see section 4.3, "Understanding PDF documents: the prototype approach") where they are placed in the context of contemporary document understanding research.

#### **3.3.3.1 Typeface comparison**

An important aspect of PDF is that all the font information used to format the text is held in the file format. Although in many cases the actual fonts themselves are not held in the document, there is a unique set of metrics for every unique font present. Analysis of these metrics provides useful information to the block segmenting routines. Furthermore, the metrics can be compared with each other in order to find out which font style is geometrically more prominent on the page. The results of this comparison

provide essential data to the algorithms which create logical associations between text blocks based on their geometric properties. Type 1 and TrueType fonts are rendered onto the screen by mathematical operators using the information the metrics hold plus the appropriate character widths. Unfortunately no such metrics exist for Type 3 fonts. Type 3 fonts are 'user defined' fonts. Typically Type 3 fonts are bitmap fonts and cannot be described by mathematical operators. No font style comparison routines exist in either the prototype system or the final system for Type 3 fonts.

The prototype system compared font metrics using the following heuristics. For any rendered text, the point size was the most important clue regarding geometric prominence, for example, text formatted in twenty point is more prominent than text formatted in twelve point. This heuristic is true in the majority of cases. However, there were cases which failed this heuristic. If the point size of text block A was formatted in a point size *just* larger than text block B but the font of text block B was artistically designed to be extremely prominent, then text block B appeared to be more prominent on the page even though it had an inferior text point size. The prototype metric did not attempt to rectify the cases which broke this heuristic. However it did have a number of primitive heuristic rules which allowed it to make a more accurate diagnosis when presented with two metrics in which the point sizes were identical.

Text formatted in bold italic text was considered to be more prominent than bold text. Bold text was considered to be more prominent than italic text. Italic text was considered to be more prominent than normal text. Sans-serif fonts were considered to be more prominent than serif fonts. The nature of the font style was determined by analysing the name of the font. A serif font was identified from the serif metric entry in the font metric table. This was a poor heuristic but it functioned well enough for the purposes of the prototype. A stronger heuristic based on more detailed metric entries was developed for the final system (see section 5.5.2.4, "Structure KS" for more details).

### 3.3.3.2 Prototype segmentation techniques

The text-segmentation strategy implemented for PDF documents was bottom-up. This was decided partly because the system was designed to analyse any class of document. No model-driven approach could have been adopted because no document model was known beforehand.

The other reason for using a bottom-up strategy was that the API (which Adobe exports with the Acrobat Exchange PDF file viewer) gives the programme developer the words of the document as logical entities. Furthermore, the API attempts to give the user the lines in the document. However, the lines cannot be accepted as being always valid<sup>1</sup>. Infrequently in geometrically complex documents the API does not find the true end of a line, for example in newspapers where the column guttering (inter column gap) is shallow and could be misinterpreted as an inter-word gap. The prototype and the final system naïvely assume that each line is valid and accepts the API definition of the line without question. It was felt that the infrequency of the API errors did not warrant either an implementation of a new text-segmentation algorithm or a complete error context identification and rectification analysis sub-system.

The input to the prototype block forming system were lines of text. The blocks were formed on a page by page basis. The lines were sorted into a vertically increasing order. They were presented to the block forming routine starting with the line occurring at the top of the page and finishing with the line occurring at the bottom of the page. Each line's geometric properties were examined and it was either added to the block currently being segmented by the system, or if no compatibility with the geometric properties of that block was made, the current block was sealed off and a new block was formed containing the current line. This algorithm was efficient when processing documents that had been formatted with a single column but further processing was required when processing a multi-column document.

---

1. This conclusion is drawn from the results of experiments conducted with Adobe Acrobat Exchange version 2.1

## Document Analysis

---

The algorithm which established whether a line was compatible with a block started its analysis by looking at the style of the line. If the line had the same point size and font metrics as the block, further analysis was made, otherwise the line was rejected. Every block in the system kept a record of the last line added to it. The position of the candidate line was compared with the last line accepted by the block. The position of the candidate line was said to be compatible with the current parameters of the block if one of the following conditions was satisfied:

- the start coordinate of the line *approximately* matched the 'leftmost' value of the block's bounding box **and** the end coordinate of the line *approximately* matched the 'rightmost' value of the blocks bounding box;
- the start coordinate of the line *approximately* matched the leftmost value of the block's bounding box **and** the end coordinate of the line was less than the 'rightmost' value of the bounding box;
- the centre of the line *approximately* matched the centre value of the block's bounding box **and** the start coordinate of the line was either greater than or *approximately* equal to the 'rightmost' value of the bounding box **and** the end coordinate of the line was either less than or *approximately* equal to the 'leftmost' value of the bounding box;
- the end coordinate of the line *approximately* matched the 'rightmost' value of the block's bounding box **and** the start coordinate of the line was greater than the 'leftmost' value of the bounding box.

The term 'approximately matched' represents a 'degree of leniency' rather than the absolute conformity which 'equals' implies. The degree of leniency was implemented as a range either side of the target value into which the argument value was successfully accepted as being 'approximately equal to'.

## Document Analysis

---

The use of the degree of leniency was extremely important to the performance of the system. For simplicity, it was decided that the degree of leniency should be a constant value. In other words the degree of leniency should not change from block to block, page to page or document to document. Initially, it was felt that the optimum value for the degree of leniency should be calculated from a function involving the point sizes of the two blocks involved. However, good results were obtained using a constant value.

The magnitude of the degree of leniency was decided upon after a series of tests on sample documents. Care was taken not to make the magnitude too large and thus risk accepting lines into blocks when the lines were not suitable (under segmentation). Similarly, care was taken not to make the magnitude too small and thus risk rejecting lines that were valid members of the block (over segmentation).

If the line was deemed to be in the correct horizontal position, an analysis was made to determine whether it was in the correct vertical position to be a member of the block. Essentially this was a test for compatibility between the existing inter-line value in the block and the vertical gap between the last line to be accepted into the block and the candidate line. Once both values have been calculated another 'approximate comparison' was made. In a manner similar to the test carried out on the line's horizontal position a degree of leniency was introduced. The magnitude of the degree of leniency was engineered based on trials made on sample documents.

The analysis of the leading position was not carried out if there was only one line present in the block. In this situation a heuristic was applied. The heuristic created a false leading value for the block. The value of the fake leading was calculated at twice the magnitude of the point size of the single text line present in the block. The fake leading value was also used as a maximum threshold limit for any line's leading value which was being analysed against the block's geometric properties. No text block would be created in which the gap between lines was more than that of twice the point size of the member lines.



In practice, this algorithm was prone to error. The prototype system calculated the inter-line gap for a block simply by looking at the gap between the first two lines in the block (if the block had two or more lines). If the inter-line gap between the first two lines was anomalous, then an anomalous inter-line gap value was set for the entire block. A better algorithm should have 'looked ahead' and analysed the inter-line gaps of the other lines on the page and not simply assigned the inter-line value for a block from the analysis of the first two lines it had seen.

The algorithm outlined in the previous paragraphs will segment a page's text efficiently if that page is single columned. For more complex page formats the algorithm may produce over segmented results, for example, in a multi-columned document in which the lines have been sorted in decreasing vertical order over segmentation will occur. Many lines will not be allowed to join the text block currently held by the segmentation routine as they are logically out of place thanks to being sorted based on their vertical position.

To counteract the potential over segmentation problem a second pass was made over the blocks created from the first segmentation pass. In the second pass, each block was allowed to look down the page and examine the block positioned immediately underneath it. This was a computationally inefficient algorithm as each block had to check the length of the list once to ensure that it found the block which was directly underneath it. The algorithm runs in  $n^2$  time, where  $n$  is the length of the block list created from the first pass. Once the block (block A) had found the block (block B) lying directly underneath it on the page, block A conducted tests upon block B in order to determine whether or not block A could merge with block B. If block A was confident it could merge, it created a link to block B. After the second pass, all blocks with links were merged together. The tests that the blocks carried out on each other consisted of testing for font and point size compatibility and inter-line gap compatibility. The inter-line gap of both blocks were compared as well as the gap between the blocks. If the gap between the blocks matched the leading of both blocks then a link from the upper block was made to the lower block.

# Document Analysis

Figure 5 shows a page of a document which has been segmented by the prototype. The page is geometrically simple. It has only one column of text and no images or graphics. Figure 5 shows the geometric blocks created from the prototype text-segmentation routines. Figure 5 also indicates the class of the blocks by using different coloured ink on the bounding boxes: maroon indicates the document title; purple indicates a text body block; green indicates a level-1-title (section) and blue indicates a level-2-title (subsection). Different inter block relationships are represented using black lines of varying thickness. The block classification techniques, the inter block relationships and the block tagging routines are all explained in section 4.3, “Understanding PDF documents: the prototype approach”.

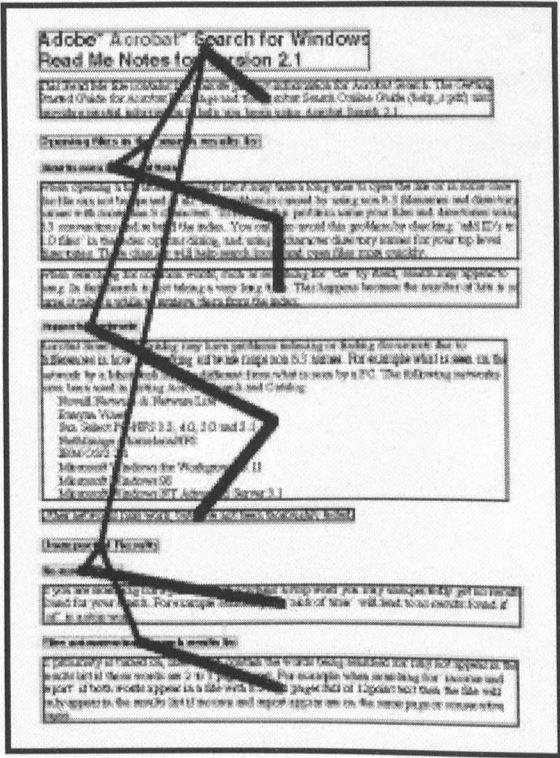


Figure 5: Typical output from the prototype PDF processing system

The TIFF bitmap decomposition algorithms outlined in this chapter provided good experience of text line and text block construction which will be shown to be productive in the context of PDF analysis in Chapter 5: “Final System Development and Engineering”. The development, research and results of the prototype PDF analyser will be shown to contribute to the development of the final system’s algorithms for PDF analysis and PDF document classification.

# Chapter 4, Document Understanding

This chapter opens with a literary survey of contemporary document understanding research. This is included to give an overview of document understanding to the reader, since elements of document understanding are present in the proposed solution to the document classification problem. This chapter goes on to give a description of a prototype PDF document understanding system.

# 4.1 Introduction

The term ‘document understanding’ is used in computer processing terminology to describe the algorithms and routines which formulate the logical entities and logical structure of a document from a document’s geometric tree.

A document image which has been geometrically segmented into regions of geometrically similar text, graphics and images, and then structured into a geometric tree that is the input for a document understanding system<sup>1</sup>. Whereas document analysis systems could afford to process one page at a time, a document understanding system must analyse the entire document as a logical entity. Document understanding must accomplish a number of critical sub goals in order to achieve an understanding of the logical document structure:

- identify the class of document being processed;
- attach a logically semantic label to the geometrical elements present in the document which is appropriate to the class of document being processed;
- identify the logical structure of the document.

All document understanding systems must know what logical class of document they are processing. Document understanding routines are document class specific. Many document understanding systems are flexible enough to process documents from the same logical class which vary in their page layout and overall design.

The class of document being processed not only dictates the set of logical tags that will be used to tag the geometric blocks in that document, but it also dictates the choice of strategies that will be used to tag the blocks with the labels and establish the logical

---

1. Some systems (notably the Graphein system [Chene91] and the system developed by Kise [Kise93]) merge document understanding into the document analysis stage of processing. Generally speaking however, document analysis and document understanding have a minimal overlap.

relationships between the tagged blocks. All document understanding systems rely upon the presence of a strong document model to help them with the labelling process and the logical structure finding process.

Document understanding systems have developed in this fashion because the logical document classification of a bitmap image is a complex problem for a computer to solve. The information that a computer needs to know in order to establish for certain the correct logical class of a document can only be found by applying model-driven document understanding routines. Document understanding systems use highly focused recognition algorithms which search for specific logical entities and specific logical relationships between those entities. Identifying logical entities and their relationships is the only guaranteed manner of identifying the precise logical class of a document.

This presents a circular argument to document understanding systems. The logical class of a document can only be determined by processing the results of document understanding algorithms. Document understanding routines can only be applied once the logical class of a document is known. Researchers in this field have circumnavigated this problem by assuming *a priori* knowledge of the class of document. Contemporary document understanding systems are designed only to process one logical class of documents. Lam calls these systems “closed” systems [Lam94a].

A priority goal of this research is to execute document understanding algorithms upon documents whose logical class is not known before hand. Whilst researching contemporary document understanding systems various shortcomings became clear. The most significant of these shortcomings was the reliance upon *a priori* knowledge of the logical class of the document image. The main attention of this research was then directed towards creating a system which would be able to find the correct logical class of a geometric document *whilst* processing that document. Chapter 5, “Final System Development and Engineering” outlines a proposed solution to the document classification problem. The solution contains elements of document understanding within it.

A literature survey of contemporary document understanding systems has been included in this chapter for several reasons. A basic comprehension of document understanding systems will highlight the problems such systems encounter. Furthermore, the role that document understanding routines perform in the final system (and the PDF document processing prototype) will become clearer.

## 4.2 Literature Survey

More research has been undertaken on document analysis than on document understanding. As the document analysis field expands and matures the natural progression of document image processing research will increase document understanding research. This literature survey is structured by author rather than by document understanding strategy.

### 4.2.1 Ishitani's Form understanding

Ishitani [Ishit95] applies a model based strategy for the specific document analysis and document understanding of forms. Watanabe [Watan93] suggests that forms are documents in which the logical structure is explicitly defined by the format of the document. Graphic lines, text columns and text rows all explicitly define the logical structure of tables and forms.

Ishitani is able to perform thorough document analysis of the ruled lines (in a form document) and then apply a specific model matching technique to find the best form model for the original document. Once the model has been correctly matched, document understanding is completed, as all elements of the image will be known and their relationships known if the model is detailed enough. The model matching problem is translated into a problem of searching for the optimal match between two structural descriptions: a formal model and the lines from the original image. Ishitani constructs

an association graph from the two structural descriptions and then searches for sub-forms and line matches.

Ishitani claims a robust and effective algorithm. This is due to the existence of very strong dominant features (lines) in the forms upon which good pattern matching techniques can be used.

### 4.2.2 Watanabe's document definitions

Watanabe [Watan93] defines four logical document classes and describes how the different geometric features present in each class dictate the optimal document understanding strategy.

- Class 1: documents have their own inherent layout models which are rigidly predefined on the document sheets. Element separators/boundaries are highly visible and explicitly defined. Examples of this class of document include forms such as those processed by Ishitani [Ishit95].
- Class 2: documents have geometric relationships among the items of the document. The relative positions of the individual items are determined according to their allocated states. Watanabe suggests that library cards are good examples of this class of document.
- Class 3: although the documents have their own layout models the relationships among individual items are determined by logical models. The meaning of each item is determined by the logical model. Newspapers and magazines belong to this class of document.
- Class 4: the documents do not conform to any layout or logical models. These documents are designed by human perception. Advertisements are examples of these highly unstructured documents.



Watanabe goes on to attempt structure recognition for the first three categories of documents. No structure recognition is made on the fourth class for obvious reasons. There follows a brief synopsis of their procedures and findings. It is worth noting that Watanabe's experiments at finding structure did not include global document classification, thus one can only presume that he applied his procedures with prior knowledge of the document class.

### 4.2.2.1 Understanding Form documents

Watanabe, like Ishitani, acknowledges that the important information in a form document is defined by vertical and horizontal line segments. The structure of a form document is physically and explicitly defined in advance. Watanabe defines this structure using a binary tree. The tree represents the logical relationships among item blocks. He defines two such trees: a global structure tree and a local structure tree.

*"The local structure tree specifies a cutting method for repeated division of the reassigned block into two rectangular blocks, starting at the upper left and working to the lower right corner. The global structure tree represents the linking method among individual rectangular blocks contained in the document, starting from the upper left corner. Nodes in the global structure tree accompany appropriate local structure trees. The global and local structure trees establish a complementary relationship to specify logically the structures of table form documents"* Watanabe [Watan93]

He then applies a document analysis technique which is based on the decomposition of the vertical and horizontal line boundaries. This stage consists mainly of binary transformations, edge extractions and corner detection procedures. The result is a set of points representing the block corners. The logical structure recognition module then interprets the connective relationships among the extracted points to identify the individual item blocks. The logical structure tree of the original image is built by examining the positions and sizes of the individual item blocks.

Watanabe uses a hybrid approach of feature extraction (bottom-up) and item-block extraction (model-driven) to analyse and understand form documents. Thanks to the explicit definition of the structure of forms, he can tell from the resulting logical tree the relationships between the logical elements without analysing the location or appearance of those elements.

### 4.2.2.2 Understanding Library Cards

Library cards are less explicitly structured and contain more implicit information in their layout than forms. As before, Watanabe defines a logical specification for the class of document being processed and, as before, he chooses a binary tree representation. The branches point out the relationships between geometric elements present on the card. Document analysis is performed using white space analysis and a geometric tree structure is built up from the analysis. Watanabe extracts the image areas (including document items) based upon their mutually neighbouring relationships rather than the semantic content of the actual document items themselves. Consequently,

*“the recognition procedure works successfully without difficulty even if there are variations in the geometric configuration caused by the lengths and numbers of items of data”* Watanabe [Watan93]

### 4.2.2.3 Understanding Japanese newspapers

Whilst processing Japanese newspapers, Watanabe defines his own physical classification of document structure as well as a logical classification. This classification has some aspects which are specific to Japanese newspapers and others which can be applied to newspapers and magazines in general. After document analysis, Watanabe is left with classified geometric blocks. The geometric blocks are said to belong to one of four classes:

- Class L: The class of horizontal and vertical separators which typically divide columns;

## Document Understanding

---

- Class P: Physical blocks such as photos, tables and figures;
- Class T: Physical blocks of articles. These blocks are surrounded by blocks of class L;
- Class S: The set of physical blocks such as titles, subtitles. Additionally, class S provides a default class for blocks which do not fit other class descriptions.

He represents the classified blocks using an adjacent relationship graph which models logical relationships between blocks. In these graphs nodes represent blocks and the edges indicate adjacent relationships. Three kinds of information are assigned to each node: class identification, coordinate values, and upper/lower column numbers. Article blocks (class T) are divided and sub classified in order to represent their detailed structure. The divided article blocks are split into text strings which belong to one of the following classes.

- NOR: the string is a standard line in which the number of characters is normal.
- PB: the string area is at the beginning of a paragraph.
- PE: the string is at the end of a paragraph.
- CEN: the string is in the centre of a paragraph.
- OTH: other – the default classification.

Watanabe acknowledges that the logical structure of Japanese newspapers is not consistent with the geometric structure.

*“The logical structure is defined by geometric relationships among the components related to individual articles and spatial relationships among individual articles on a page. They are different from page to page” Watanabe [Watan93]*

### Production rule expert system

In newspapers, the page layout is defined by the number of articles and their size and their importance as well as the mutual relationships among articles. To accommodate this flexibility Watanabe uses a production rule system to recognise the logical structure of a page. He devised two sets of rules. Rule set two interprets the logical structure of the input page maintained by the adjacent relationship graph. Rule set two invokes rule set one to interpret the detailed structures maintained by the characteristic relationship lists under control of set two. Watanabe interviewed a newspaper editor in order to compose these rules.

Rule set one is invoked to determine whether a physical block is a title, a subtitle or part of an article by examining its detailed structure. In the following rules A represents the block of interest. Th is a threshold value used to judge the height to width ratio of titles and sub-titles.

[Title block] If A is class S and its dimensional ratio is greater than Th and it covers two or more columns then A is a Title block.

[Text block] If A is class T then A is a Text block.

[Head block] If A is class T and its first string belongs to either CEN or OTH then A is a Head block.

A Head block is a text block containing a minor title which fits in a column.

[Begin block] If A is class T and its first string area is attached to PB then A is Begin block.

This rule determines the beginning line of a paragraph in the physical block of class T.

Rule set two is used to merge individual physical blocks into a news article. The variables in square brackets refer to the activation of rule set one. Current points to the current physical block and Number keeps the label of a physical block.

## Document Understanding

---

If A is [Title block] and its left node is [Begin block] and Current is NULL then add(Number), label (A), label (A's left node), move Current (A's left node).

Once a title has been identified, the above rule tries to determine whether its left adjacent node contains the first paragraph of the article or not.

If A is [Text block] and its right node is the same [Title block] as Current and A corresponds to the next column of Current, then label (A), move Current (A).

Since a text block that is left of and adjacent to a title belongs to the article containing the title, the rule stated above merges the block A into the article.

If Current is [Text block], then begin find next (Current); if there is a next, then label (next), move Current (next), else NULL (Current); end.

This rule is used to merge an article with the current article. find next searches the remaining columns in to find the next article block. It achieves this with a three step algorithm.

- Step 1: search every edge from current to find the first nodes of class T.
- Step 2: find the first unlabelled node in the class T.
- Step 3: check the characteristics of current and the newly found node for consistency. If the last string in current is labelled as PE and the first string of the new node is labelled as NOR then they are incompatible. Otherwise, the new node is the continuation of the article to which current belongs to.

If [Head block] and Current is NULL, then add (Number), label (A), move current (A).

If A is [Text block] but neither [Head block] nor [Begin block], and Current is NULL, then backtrack.

## Document Understanding

---

If a major title is not used to start an article then either a minor title (Head block) or an indented text line (Begin block) is used. Backtracking finds all nodes which may possibly correspond to A's previous text sections. If backtracking fails then the logical structure recognition finishes.

### The control strategy

The execution of the document understanding process is driven by the knowledge of the composition rules for newspapers in the form of the production rules. The basic recognition process has three phases.

- Step 1: match the rules of rule set two to the adjacency graph constructed from document analysis. The selected rules are held as competitive rules.
- Step 2: select the first applicable rule. The first rule activated from the order in which they were described previously has the highest priority. In cases of equal priority the rule involving the block which is closest to the upper right hand corner of the page is selected.
- Step 3: execute the right side of the selected rule. The adjacent relationship graph is modified according to the interpreted results.

This is repeated until all the rules are exhausted or the back tracking fails. The logical document tree composition process is controlled by a top-down or model-driven approach.

By looking at three classes of document, which vary in degrees of geometric and logical structure, Watanabe has shown that documents which are geometrically structured (table forms and to a lesser degree library cards) can be logically represented in a binary tree structure and that his framework for document understanding works well in both cases. Newspaper documents are not as constrained in their layout as either of the other classes and consequently a flexible document understanding strategy is required.

Watanabe suggests that the concept of multi-level recognition layers can be applied to various classes of documents. The main distinction between documents, he states, is

*“...derived from the adjacent relationships among different item blocks, allocated in two dimensional space”* Watanabe [Watan93]

The basic knowledge representation tool in all of these strategies is knowledge about the document logical structure. The importance of recognising the class of document being processed is paramount in order to gain this knowledge and direct the correct logical document understanding strategy.

### 4.2.3 Niyogi's newspaper understanding system

Niyogi and Srihari have also developed a computational model for document understanding in which a rule based control strategy is employed. A hierarchical rule based system is used to guide block classification, grouping and reordering operations, which is used in conjunction with a domain knowledge base which encodes rules governing document layout [Niyog95, Niyog96]. Heuristics are used to infer the classes and labels of the blocks and additionally the reading order of the blocks is inferred. Niyogi only processes newspaper type documents. This class is geometrically diverse and contains many layout rules and guidelines and may be considered as complex geometric documents. Once again, by knowing beforehand the class of image to be processed, model-driven processing can be utilised for document understanding.

Niyogi uses a top-down, rule based backward chaining strategy for document processing. He uses O'Gorman's docstrum plot segmentation algorithm for document analysis which is appropriate considering the complex geometry of the class of document being processed (see “Document Spectrum Plot” on page 41 and section 3.2.2.4, “An assessment of Page Segmentation strategies” for more details). Niyogi gathers as much information about the geometric characteristics as he can from the

# Document Understanding

document analysis stage. By examining the block size and the connected components with his segmented zones he estimates the type of text in those zones and so builds up a more detailed picture of the geometric document structure. Table 6 and Table 7 show Niyogi’s definitions of a newspaper’s physical syntax and logical syntax respectively in Extended BNF notation.

|            |     |  |
|------------|-----|--|
| <document> | ::= | { <page> }   |
| <page>     | ::= | { <block> }  |
| <block>    | ::= | <large-text>   <medium-text>   <small-text>   <line-drawing>   <half-tones>   <boundary> |
| <boundary> | ::= | <horizontal-line>   <vertical-line>   <line-rectangle>                                   |

Table 6: Niyogi’s physical newspaper structure

|                  |     |   |
|------------------|-----|---|
| <document-info>  | ::= | { <unit> }  |
| <unit>           | ::= | <title>  <br><graphical-area>  <br><story>  <br><photoblock>  |
| <photoblock>     | ::= | [ <title> ]   <photo> <caption>   |
| <graphical-area> | ::= | <page-banner>  <br><horizontal-band>  <br><other-graphics>  |
| <story>          | ::= | [ <sub-story> ]  <br><title> [ <sub-title> ] { <text-para> }<br>[ <photoblock> ]<br>[ [ <title> ] <table> <caption> ] |
| <sub-story>      | ::= | <story>   |

Table 7: Niyogi’s logical newspaper structure

Niyogi acknowledges that layout rules vary widely among different types of documents and even among samples of the same type of document.

*“Thus, a knowledge base of layout rules for document logical structure derivation will contain some global rules that apply to a majority of documents and some domain specific rules that apply only to the type of document being analysed”* Niyogi [Niyog95]



# Document Understanding

In the case of newspapers a domain specific rule may be that multi-line headlines are left-justified, or that a thin line rectangle around a set of blocks signifies an independent story. An inference engine within the rule based system makes deductions about the document using a hierarchical knowledge base that contains rules about identifiable characteristics of document images. Niyogi's rule system closely follows other well documented rule based strategies in that three levels of rules are implemented: knowledge rules, control rules and strategy rules [Winst92, Jacks92].

Knowledge rules contain all the domain knowledge for the system. All common characteristics of different types of document blocks as well as spatial constraints commonly followed in document layouts (for example the positioning of captions relative to images) are encoded here. These knowledge rules can be used for block classification, block grouping and text block ordering as and when directed by the control strategy. An example of Niyogi's knowledge rules is show in Table 8 (a).

|  |
|--|
| <p><b>IF</b> a block Z is of type "large-text"</p> <p><b>ELSE IF</b> it satisfies the following three conditions:</p> <p>{</p> <p>    it is of type "medium-text",</p> <p>    <b>AND</b> it is below another block W,</p> <p>    <b>AND</b> block W is not of type "large-text" or "medium text"</p> <p>}</p> <p><b>THEN</b> block Z is a major headline.</p> <p>(a) An example knowledge rule</p> |
| <p><b>IF</b> the grouping mode is on,</p> <p><b>AND</b> a block has been selected,</p> <p><b>THEN</b> find all the immediate neighbours of the selected block</p> <p>(b) An example control rule</p>   |
| <p><b>IF</b> any partially grouped units remain,</p> <p><b>THEN</b> apply all unit-related control rules for each of these units until there are no more partial units</p> <p>(c) An example strategy rule</p>   |

Table 8: Examples of Niyogi's rules from different knowledge hierarchies

Control rules regulate the invocation of the knowledge rules based on appropriate processing states. An example of Niyogi's control rules is shown in Table 8 (b).

Strategy rules are general meta-rules. Strategy rules determine what control strategy is to be followed for analysing the image. An example of Niyogi's strategy rules is shown in Table 8 (c).

Niyogi's system is flexible enough to allow the addition of more knowledge rules to facilitate the analysis of additional document features. Yet this is the system's weakness. In order to be thorough the system must build up and maintain a comprehensive rule base. Furthermore, for the system to be truly universal another layer of functionality must be added to the rule base: document classification.

### 4.2.4 Dengel's business document processing

There are an infinite number of formatting variations within instances of any document class. There are also characteristics which remain consistent. Andreas Dengel and Frank Dubiel [Denge95] have studied the class of business letter documents and noted that

*"Although there is a common understanding how typical newspapers or business letters look like, information presentation as well as its logical composition is somehow fuzzy"* Dengel [Denge95]

In order to overcome this fuzziness problem, Dengel identifies logical attributes within business letters and clusters them into abstract structural concepts with which he constructs a decision tree classifier. The level of abstraction is higher than that which Niyogi utilises, and so Dengel can afford to construct more a detailed decision tree, for example, Dengel can construct a concise set of logical labels with which to tag elements: body, recipient, logo, salutations, footnote, subject, date, relation, sender and signature. Dengel is able to achieve this level of detail because business letters are highly

geometrically structured. There are standards and rules for laying out business letters, and these layout guidelines are important aids when determining structure.

Dengel's attributes are renamed distinctive marks and are determined in such a way that each geometric object located has at least one distinctive mark and that the number of distinctive marks is as low as possible. The decision tree is constructed by using a training set of forty business class documents and tested with a further forty business class documents. This potential solution to the infinite instances problem seems positive, but can only be applied with some confidence to one class of documents.

### 4.2.5 Semantic net strategies

Bayer and Walischewski have also researched structurally decomposing business letters [Bayer95]. They choose to model business letters using semantic nets which describe geometric properties, spatial relationships, lexical entities as well as lexical relationships. This work stems from original research by Bayer which uses a semantic net language (FRESCO) which is specifically designed to model knowledge about structured documents and analysis techniques [Bayer93].

Their set of labels is not as detailed as Dengel's but it does contain all the fundamental tags. From a set of one hundred and eighty one documents they trained their system with twenty documents and achieved an average rejection rate of 0.4. Bayer et al. state that the major advantage of their system is the decreased training time required by the system, although this could be seen as simply a trade off between time and detail when comparing this method with Dengel's system.

### 4.2.6 Taylor's document classifier

Taylor et al. have attempted to classify any image by document type [Taylor95]. They suggest that reasonable classes of documents might be: newspapers, business letters and technical journals.

*“A document classifier plays two important roles in a document analysis system: one to find particular class or classes of documents and two, to sort all documents into classes.”* Taylor [Taylor95]

Their general classification process is a two layer hierarchy designed to classify multi-page documents from their first few pages. Firstly, the document is sorted by the number of columns and secondly, logical components are detected to determine the class. The column detection stage is used to reduce the number of classes to be searched through by the functional classifier, for example, it is very rare for a business letter to have more than one column and equally rare for a newspaper to have less than two columns. Taylor states that logical features such as ‘salutations’ and ‘inside address’ are useful in classifying business letters from other classes.

However, there are many question marks surrounding this method, not least of which is the method of obtaining the logical features. Taylor admits that most of the rules for determining logical features are content based. They require natural language processing (NLP) technology. This wastes all the geometric information available such as the spatial relationships and element attributes. Yet, the identification of detailed logical elements such as ‘salutations’ (using geometric information only) would require *a priori* document class knowledge. Consequently, in order to avoid this “Catch22” situation Taylor has been left with no alternative but to reject all geometric information and utilise content based heuristics which are far more susceptible to misinterpretation and ambiguity without geometric information confirmation.

In her paper, Taylor starts by claiming that newspapers, journals and business letters are suitable classes to classify into, but they only present a classifier which has been designed for business documents such as letters and memorandum. Presumably this is because content based information is not good enough to determine logical elements in newspaper type documents.

Taylor's classification algorithm combines knowledge of a document's page layout with knowledge of a document's logical structure and her method of processing the semantic content of blocks in order to establish their logical role is fast but prone to misinterpretation.

The semantic understanding of text by a computer is an area of document processing and artificial intelligence which can provide vital clues to a document's logical class and structure. However, natural language processing (NLP) is an enormous field of research which has not yet matured. Even if NLP could produce valid interpretations of a block's semantic content, NLP alone would not be enough to establish the logical structure of a document. The recognition of a document's logical structure demands the analysis of logical entities and the relationships between those logical entities.

### 4.2.7 Esposito's logical rule base

Esposito et al. have approached the problem of document understanding from a Prolog style rule acquisition system [Espos93]. With a training set of twenty documents together with initial absolute truth user input, their system attempts to learn rules which will recognize logical components of instances of the same document class. It is worth noting that Esposito et al. acknowledge the importance and position of document classification within the architecture of document image processing. They describe their document classification process as

*"... given a set of classes in which documents handled in a specific office can be grouped, we use a process of supervised inductive generalization in order to generate classification rules from some significant documents of each class."* Esposito [Espos93]

A representation language (LO) is defined which describes the geometric elements present in the document image. Additionally, Esposito provides LB (a language of background knowledge), LH (a language of hypotheses), O (a set of examples or

observations described by LO) and B (some background knowledge described by LB). The goal is to find hypothesis H, described in the language LH which is a subset of pure Prolog. Table 9 (a) has examples of predicates used in LO. Esposito uses FOCL to learn the hypothesis defined by the language LH. FOCL is an extension of FOIL which is a learning system that implements a divide and conquer strategy to learn a rule or hypothesis. FOCL allows predicates to be defined intentionally and in this manner it provides a way to introduce inference rules as background knowledge to use in the induction process. For the application of document understanding, Esposito has defined several rules concerning the position of a block, the type of alignment between blocks and the mutual position of blocks. Examples of these rules can be seen in Table 9 (b).

|   |
|---|
| width-very-very-small(X)<br>height-large(X)<br>type-text(X)<br><br><b>(a) Predicates from the observation language LO</b>   |
| sender(X) ← above(X,Y), type-picture(X,Y)<br>aligned-left-column(X,Y) ← aligned-only-left-column(X,Y)<br>aligned-left-column(X,Y) ← aligned-both-columns(X,Y)<br><br><b>(b) Inference rules from the background knowledge language LB</b> |

Table 9: Examples from Esposito’s knowledge languages

Esposito et al. trained their system with twenty documents from the business letter class. After two trials the system required some background knowledge in order to improve the generalizations of some of the concepts. Although Esposito states that background knowledge alone does not significantly improve the predictive accuracy of the whole set of rules, she claims that better results can be achieved using “hierarchies of concept dependencies” based upon the spatial contiguity of logical components, for example, consider the logical “date” element on a business letter. The presence of this component is dependent upon the presence of a logical “sender” element.

## 4.2.8 Saitoh's text area ordering system

Saitoh et al. [Saito93] describe a system for document image segmentation and text area ordering. Although this system is not pure document understanding it is worth examining because Saitoh sets out to accomplish exactly the same goals as those attempted by the PDF prototype document 'understander' (described in the next section): classification of text regions and an ordering of these regions into a reading path.

Saitoh uses well established image segmentation techniques based on Pavlidis's white space analysis segmentation algorithm [Pavli91]. Saitoh classifies text areas into headers, footers, captions and text bodies. He uses the ordering of the blocks to create an ODA type layout tree [ISO89]. Saitoh segments graphic lines and uses them to help define blocks and create the reading paths through the document text blocks.

Text areas are connected together based on their *influence range*. He constructs a tree from the blocks based on these ranges. He loosely defines an influence range as being the width of a text block. If a node (a text area) is just underneath another node it is assigned as a son and the son inherits the influence range of the father node. If the influence range of a child is wider than its parents, the influence range is extended. Figure 6 is a replication of a figure that Saitoh uses to clarify his definitions.

The nodes in the geometric tree constructed by this system are bubble sorted and the reading order is extracted by traversing the tree from left to right in a depth first manner.

### 4.2.8.1 A discussion of Saitoh's approach

No algorithms are specified in his paper and as such it is difficult to gauge the accuracy of Saitoh's results. The goal of the system was to find a reading path between blocks. This is accomplished by ignoring logical title blocks and simply extracting "peripheral" text blocks such as headers, footers and captions. If these blocks were to be included in the text ordering they would break up the flow of the document. The peripheral blocks

4.2.9 Lam's adaptive reading framework

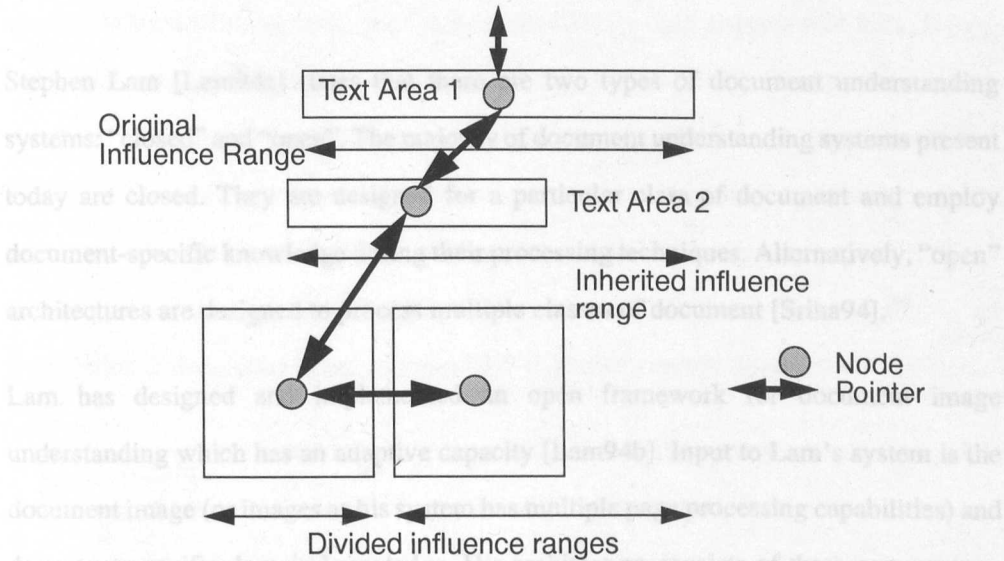


Figure 6: The influence range of Saitoh's blocks

are attached to the root node of the page on which they reside. One can then traverse the tree from the root without encountering any peripheral blocks until one returns to the root node.

Saitoh's system seems robust. He has designed it in such a fashion that it will process documents with simple page layouts (single column layout for example) and complex documents (documents with multi-column page layouts and a large number of images which break up the text flow). Nowhere is the problem of document classification mentioned although the classification problem does lie outside the bounds of text region ordering. However, Saitoh makes a number of remarkable statements which question the necessity of extracting non-textual data and the necessity of 'OCR-ing' characters which are over 1cm in height. From a human cognition aspect, it is these features which attract the most cognitive attention, and arguably, which help humans navigate through a document.



## 4.2.9 Lam's adaptive reading framework

Stephen Lam [Lam94a] states that there are two types of document understanding systems: "closed" and "open". The majority of document understanding systems present today are closed. They are designed for a particular class of document and employ document-specific knowledge during their processing techniques. Alternatively, "open" architectures are designed to process multiple classes of document [Sriha94].

Lam has designed and implemented an open framework for document image understanding which has an adaptive capacity [Lam94b]. Input to Lam's system is the document image (or images as his system has multiple page processing capabilities) and document-specific domain knowledge. His architecture consists of three components: control, knowledge base and tool box [Sriha92].

The tool box contains a set of generic document image processing tools that are applicable to different documents. Tools developed for different conceptual levels are coordinated by the control.

The knowledge base consists of two sub-components: document models and general knowledge. A document model describes the aspects of a document domain or a group of documents that share similar layout structure. General knowledge is shared by different document domains. It describes the tasks that are needed to locate and identify document components, such as text blocks and line segments.

The control selects the use of tools and is responsible for the intelligent combination of data extracted from document sub-areas to generate a representation of the scanned document. It examines the problem state in its working memory and uses the facts in the knowledge base to determine which modules in the tool box should be used.

Lam's system can perform document analysis, document classification and document understanding on four different document classes: postal mail pieces; forms; bills and journals [Lam94a]. This is a significantly more open system than Niyogi's which was designed to handle the layout variations amongst broadsheet newspapers.

## Document Understanding

---

All the documents which Lam's system processes have radically different layout models. Watanabe [Watan93] and Pavlidis [Pavli92] would suggest that bills, forms, and postal mail pieces all define their logical structure explicitly through their layout structure. There is little variation between one document instance in these classes and another in terms of appearance. Journal structure is not explicitly defined through its appearance but there is significant information available to classify a journal if, as Lam's system does, natural language processing (NLP) is used as a source of information. Information obtained from NLP is termed content interpretation by Lam.

Lam's system is very much top-down or model-driven. The system executes a model-driven classification process which is part of the document understanding technique. The document understanding problem is reduced to a constraint satisfaction problem. The control selects tools and strategies for processing the document image to satisfy constraints defined in the knowledge base. If one particular tool or strategy fails (by not meeting a constraint) then another is selected. The control portion of the system utilizes the knowledge base to select and re-select tools and strategies.

Model-driven document understanding is only as good as the models which are used to guide it. There exist certain classes of documents for which the models (which are necessary to describe the document in enough detail to perform model-driven document understanding) become extremely complex, for example, the document models which define the appearance and structure of magazines and broadsheet newspapers are going to be very similar at an abstract level. The models would have to be highly detailed and contain a deep level of logical and geometric information before significant differences become identifiable. The more complex the document models become the more complex the strategies used to detect their features become and the more detailed the knowledge base needs to be. Lam's approach, therefore, is well suited to documents with simple geometric models, in other words, documents with simple layouts.

## 4.3 Understanding PDF documents: the prototype approach

The remainder of this chapter is devoted to the description of a prototype PDF document processing system, developed whilst the author was working as an ‘internal student’ at Adobe Systems Inc.

Adobe required a program which would help make their PDF file format more accessible to people with visual impairments. Adobe rationalised that the creation of text blocks and the creation of a reading order for these blocks were the two priority goals. In fact, a disabled user who downloads a PDF document would benefit greatly from *complete* computer document understanding of that document and not just the extraction of blocks and the creation of a reading order between those blocks [WWW96]. This is a scenario in which a logical document classifier would be invaluable. The disabled user would download a PDF document, engage a document processing system which would analyse, classify and logically understand the downloaded document and then have the resulting logical document read to them by a screen reader.

The development of this system marked the transition of the author’s interests from document analysis to document understanding. The prototype was used as a platform to test and evaluate document understanding routines, although this was given a lesser priority than the two main project goals.

Before attempting document understanding, the prototype segmented the PDF document using analysis techniques which have already been described in section 3.3.3, “Basic Document Analysis of PDF” in the previous chapter.

The discussion of the prototype has been included after the literary review in order to highlight some of the issues and aspects which face contemporary document understanding systems before examining an experimental system.

The prototype attempted a very basic level of document understanding. It could only apply document understanding routines to two classes of document: newspaper documents and academic documents. It did not use any document classification heuristics and instead “asked” the user via a dialog box to identify the class of document being processed. The system achieved a basic level of understanding by finding simple logical associations between blocks. The nature of these inter block associations is different for each class of document [Watan93].

The process of classifying blocks and finding associations was very much a symbiotic process. The logical associations were made based on information provided from the basic geometric properties of the blocks and block tag information; the block tagging process used pattern finding routines, geometric information and logical information. The development of this symbiotic relationship was a major influencing factor in deciding to implement the final system with a blackboard framework (see section 5.4.1, “Blackboard systems” for a definition of a blackboard framework) at the core of the final system model (outlined in the next chapter). A blackboard framework can formalise the flow of information between the two document structure models (logical and geometric).

### 4.3.1 Classifying blocks

The prototype had to deal with the problem of tagging geometric blocks from a wide variety of logical document classes. Two proposals were considered as possible solutions:

**Proposal 1:** store a database of detailed logical tags together with the conditions for their application to a geometric block.

This proposal would provide an accurate level of document understanding thanks to the specific nature of the tags. Unfortunately the overhead required to implement this proposal as a practical working prototype was immense. The database of tags would

need to have routines to update and maintain it. The system would need to have routines to acquire document-specific knowledge from examples of classes of documents and routines to store this knowledge as rules in the database. This option was not taken up due to lack of time for developing the prototype whilst at Adobe.

**Proposal 2:** create a basic set of tags which one would expect to find in the majority of documents and use these tags to help create logical relationships between geometric blocks in the document. Table 10 lists the tags used by the prototype system. The set of tags is a mixture of logical tags (for example titles) and geometric tags (for example header and footer).

|  |
|--|
| <b>Main Text.</b> This tag represents the blocks in the document which contain the document's content  |
| <b>Title 1.</b> This represents a title block. A structural hierarchy of title blocks can be achieved by utilising the other two levels of title tags when tagging blocks. |
| <b>Title 2</b>   |
| <b>Title 3</b>   |
| <b>Super Title.</b> This tag represents the title of the document.   |
| <b>Header</b>  |
| <b>Footer</b>  |
| <b>Unknown.</b> Tags which could not be confidently classified were tagged as 'unknown entities'.  |

Table 10: A list of the tags used in the prototype

The strongest criticism of this proposal is that by using a set of tags from a low level of data abstraction the level of document understanding achieved by the system will be at a low level. On the other hand, this weakness is also this proposal's advantage: the tags can be applied to documents from different classes because the set of tags is not always document class specific.

In certain cases, notably titles, the blocks are true logical entities. This is because the geometric bounding box of these blocks encloses the entire logical block. This is not the case for main text blocks. All text blocks are formed based upon the geometric

properties of the text they contain; they are true geometric blocks. Most blocks tagged as ‘main text blocks’ are not entire logical entities but fractured logical entities. Part of the document understanding process is to reconstruct the fractured entities. In all ‘real world’ documents there exist a certain amount of logical entities which are spread over a number of pages, for example, newspapers typically have logical articles which are spread over two or more pages. Pages are geometric elements not logical elements. Consequently, any logical newspaper article which is spread over two or more geometric pages *will* be fractured. No cross-page logical entity building is attempted by the prototype partly because the level of document structure required by the prototype did not demand it and partly because finding cross-page logical entities is an extremely complex problem. Furthermore, the prototype does not attempt to split and re-group ‘main text blocks’ into logical paragraph blocks. The prototype treats each ‘main text block’ as a ‘pseudo’ logical paragraph because for the limited purposes of the prototype there was no reason to construct a ‘perfect’ logical paragraph.

The following sections provide a brief description of how the prototype classified and tagged the geometric blocks it created.

### 4.3.1.1 Finding main text blocks

The prototype created a histogram of all the unique font and point size text style combinations and found the mode text style value. Any block which has the same style as the mode value was tagged as a ‘Main Text’ block. This was a very simple heuristic, yet it was very successful.

### 4.3.1.2 Finding peripheral blocks

Header and footer blocks are both types of ‘peripheral’ blocks as they are always located to the sides of the logical content of a geometric page. The prototype peripheral block tagging algorithms used a ‘train and tag’ system. The algorithms used a couple of simple heuristics to locate candidate peripheral blocks with which to train themselves:

- blocks must be geometrically located within a certain distance of the horizontal page boundaries. The distance threshold was established by trial and error testing by the system implementor;
- blocks can only contain one or more text lines.

Once all the candidate peripheral blocks have been collected the prototype executed a simple pattern finding routine to locate recurring geometric features amongst the candidate blocks. The pattern finding routine looks at the horizontal position of the blocks and their text style.

The pattern finding routine found the three most frequent unique position and style combinations from the candidate blocks it was presented with. Three was thought to be the maximum amount of peripheral blocks that would occur as either headers or footers; one unique entry roughly corresponding to a left, right and centre justified header or footer block. The prototype estimated the most frequent unique location and style combinations by creating another histogram and extracting the three most frequent entries. Subroutines were used to check for exception cases, such as there not being three unique location and style combinations, or one of the unique combinations having such a small frequency value compared to the total number of pages that it was highly unlikely that that particular block was a peripheral block.

### **4.3.2 Tagging title blocks and finding logical dependencies between blocks**

Main text blocks, headers and footers were all tagged before the prototype tagged title blocks or made any dependency associations between blocks. No logical block dependency knowledge was required to tag main text blocks as the prototype's logical model prevented main text blocks from making associations to other non-main text blocks. Main text blocks could be dependent on other blocks, that is they could have links made to them.

Headers and footers are pure geometric entities and as such cannot be logically dependent on other blocks or have other blocks logically dependent on them.

The extraction of peripheral blocks and main text blocks from the set of all geometric blocks left a set of ‘unknown’ blocks. The set of title blocks is a subset of the set of unknown blocks.

### 4.3.2.1 Finding title blocks

Title blocks were tagged by creating and analysing logical associations between certain blocks and then analysing and comparing the geometric styles of the potential title blocks. The logical associations were created using an algorithm which consisted of two separate stages. Firstly, the algorithm made general associations between blocks based on geometric properties. Secondly, the class of document was given to the algorithm and class specific subroutines were engaged to tag the title blocks and create final logical associations.

### Making basic dependencies between blocks

The prototype created two types of dependency between blocks. The first class of dependency existed between main text blocks only. This dependency finding routine was an attempt to partially rectify the fractured logical entity problem outlined earlier in this chapter. In summary, main text block A was allowed to link to main text block B, if block B was located directly below block A and the horizontal coordinates of the bounding box of block B were approximately equivalent to those of block A. This technique was similar to the block merging algorithm outlined in section 3.3.3.2, “Prototype segmentation techniques”, except there was no analysis of the leading properties of the two blocks and no block merging was performed. It was found that the dependencies created by this routine were very secure and could be considered as document class independent dependencies.



# Document Understanding

The second type of dependency was made between unknown blocks and either other unknown blocks or main text blocks. This type of dependency was not class independent and had to be revised once the prototype knew the class of document being processed. However, this algorithm did provide essential information to the prototype about candidate title blocks and the nature of the dependencies that stemmed from those candidate title blocks.

An unknown block (block A) was linked to another block (block B) using the rules outlined in Table 11.

|   |
|---|
| <b>Rule 1:</b> Block B cannot be a member of either the header or footer block class.   |
| <b>Rule 2:</b> There must exist a line which perpendicularly intersects both the base line of block A and the base line of block B without intersecting any other block's base line. The 'base line' of a block is the bottom line of the block's bounding box. |
| <b>Rule 3:</b> Block B must be of a lesser geometric style. See section 3.3.3.1, "Typeface comparison"  |

**Table 11: Rules for linking block A to block B**

If a block B became logically dependent on block A, then block A was tagged as a title block. The prototype did not know which specific title tag to use as not all of the candidate title blocks had been processed. No judgment could be made upon the hierarchy of the title blocks until all the candidate titles had been processed.

After extracting the set of all titles from the set of unknown blocks, the prototype made a guess at the title of the document. Generally speaking the title of the document lies on the first page of the document and is formatted in a text style which is the most predominant throughout the document. Furthermore, that text style is not found on any other pages in the document.

In order to continue document processing with class specific document analysis algorithms the prototype 'asked' the user to identify the class of document. The prototype presented a choice between "newspaper" documents and 'non-newspaper' documents. The class 'newspaper' was loosely used to describe any document whose

logical structure was represented by a series of logical articles, for example, newspapers, magazines and newsletters. The class 'non-newspaper' was loosely used to describe any document whose logical structure was composed of chapters, sections and subsections, for example, books and academic journals. These definitions are too simple to be useful to normal document understanding systems but they meet the demands of the prototype.

### Newspaper specific document understanding algorithms

The document understanding routines used to process a newspaper attempted to isolate logical articles throughout the document. The prototype defined an article as consisting of a title block and any combination of main text blocks and sub-stories. A sub-story consisted of a title block and some main text blocks. No strict document definition grammar was used to help construct the logical document because the prototype was attempting to force a wide range of logical document classes into one logical class. The use of a grammar would have been too restrictive. One heuristic which was strictly applied was that the sub-story title block should have a lesser geometric style than the article title block style.

An important difference between the document understanding routines used by the newspaper understanding algorithm and the academic understanding algorithm lies in the procedures which analyse the differences between title blocks in the two document classes. In academic documents the geometric differences between title blocks are calculated and the differences convey information regarding the logical status of the blocks. Subsection title blocks always have a lesser geometric prominence than section title blocks. In newspaper documents there are many different text styles for logically equivalent entities. There is a great deal of inconsistency in newspaper documents between article title block styles, although they are all logically equivalent. Typically the differences are deliberate because the designer of the page wanted to ensure that one article on the page (the headline article) was read before any articles which the designer considered semantically inferior. The positions of articles on the page of a newspaper dictates to the reader in which order he or she should read those articles, rather than any logical hierarchy between articles (assuming that all the articles are logically equal).

## Document Understanding

---

The differences in the relationship between the logical roles and the geometric properties of the title blocks in both newspaper documents and academic documents are realised in class specific variations of the typeface comparison algorithm. The academic document understanding routines used the algorithm outlined in section 3.3.3.1, "Typeface comparison". This is a straight comparison algorithm which compared the geometric prominence of two blocks. The newspaper document understanding algorithm adjusted the original algorithm so that it was more relaxed in its comparison. The result of this leniency was that two title blocks which were approximately the same size were considered logically equal in the context of newspaper document understanding. A title block (block A) would only create a logical dependency to another title block (block B), if block A was *substantially* more prominent on the page. In this context block B became a sub-story title block.

The input to the document-specific prototype document understanding routines was a set of tagged blocks together with a set of logical dependencies between the blocks. The newspaper document understanding algorithm sorted these blocks into a vertically increasing order (the first block is the lowest, the last block is the highest) and proceeded to create logical dependencies amongst the blocks by examining the block's tags, position and previously defined logical dependencies. Any main text blocks which had not had a dependency created between themselves and a title block, searched up the page for the first occurrence of a title block. A logical dependency was created between a title block (block A) and a main text block (block B), if there was a vertical line which perpendicularly intersects the base line of block A and the base line of block B without intersecting the base line of another title block located between the baseline of block A and the baseline of block B.

The results of the dependency-creating pass was a dependency tree. The root of the tree was a node representing the class of document. Branches from the node represented individual dependencies to article nodes, 'unknown' nodes and peripheral nodes. The article nodes were ranked in order in which one would normally expect to encounter

them by reading the newspaper from cover to cover, starting at the first page. Within the article nodes were branches to title nodes, main text nodes and sub story nodes. All branches within article nodes were sorted into a reading order in a similar manner to that presented by Saitoh [Saito93].

The reading path of the document was found by traversing the tree left to right in a depth-first manner. The prototype re-tagged the titles in the document by traversing the tree in the manner described above. When the prototype encountered an article it tagged the title of that article as a level-one-title. All the sub-story title blocks within that article sub-tree were tagged with a title tag which was one logical level lower than that of the logical title level of their parent block, for example, an article would be tagged as level-one-title, a sub-story within that article as a level-two-title and a sub-story within that sub-story as level-three-title (and so on).

### **Academic specific document understanding algorithms**

The routines used to establish a logical structure in academic documents operated in an opposite manner to those used to process a newspaper document. Whereas the final logical dependencies between blocks were created in the newspaper understanding routines before the final block tags were set, in academic document processing the opposite took place. The formatting rules were much stricter in academic documents. Blocks which represented a certain logical entity, for example a section, were formatted in a style which was consistent for all similar logical section blocks throughout the document. Analysis of the styles of title blocks took place before the final logical dependencies were found in order to take advantage of the consistency of the relationships between the logical status of a block and its geometric text style.

All the blocks which were tagged as title blocks in the first pass of the understanding algorithm were extracted and compared with one another. A hierarchy was established between the title blocks which was based on strict comparisons between the text styles of the title blocks. Given that the prototype already 'knew' the style of the document title (or super title) and could therefore extract that style from the hierarchy structure, all

## Document Understanding

---

blocks formatted with the highest text style in the hierarchy were tagged as level-one-title. All blocks which had a text style corresponding to the next two text styles in the hierarchy were tagged as level-two-titles and level-three-titles respectively.

For each page in the document, the blocks were sorted into a increasing vertical order and the dependencies amongst the blocks were found using the following rules as search parameters.

- Main text blocks were allowed to search up the page and form a logical dependency with either another main text block or a title block. ‘Title block’ refers to any block which is tagged with one of the three levels of title tags. The rules for forming a dependency between main text blocks were identical to those used by the newspaper understanding algorithm (see “Making basic dependencies between blocks” on page 100).
- Title blocks were allowed to search up the page and form dependencies with other title blocks. In a similar fashion to the newspaper title dependency algorithm, there had to exist a vertical line which intersected the base lines of both blocks and no other title blocks’ base lines.
- No title block was allowed to form a logical dependency with another title block which was tagged with an identical title tag.
- No title block was allowed to form a logical dependency with another title block which was tagged with a title tag that existed on a logically inferior level in the title hierarchy.

The resulting logical dependency tree was parsed to ensure that a consistency amongst dependencies was present, for example if a level-three-title tag was dependent on a level-one-title tag the prototype would examine the context of this dependency because it would expect a level-three-title block to be dependent on a level-two-title block rather than a level-one-title block.

### 4.3.3 Prototype document understanding discussion

The PDF document prototype provided an excellent platform to test many of the algorithms which were refined, modified and improved before their inclusion in the final system. The most important lesson learnt was that the level of document understanding achieved by the prototype was far too ‘general’ to be of any practical use in a logical document database. This shortcoming was a direct result of not knowing the specific class of document being processed. However, it was noticed that this general level of understanding could be used to help extract meaningful document features which could help to classify the document.

The implementation of the prototype also provided experience with processing PDF documents. The prototype was implemented over three months and during its development it was clear that although certain avenues of research would be achievable, they would be wasteful in terms of time spent developing them and the overall impact they made on the performance of the prototype, for example, the handling of the geometrically invalid text lines that the Acrobat Exchange API can give to the prototype and the poor detection of hanging titles. Hanging titles are paragraph titles which are formatted in a second column to one side of the body text of which they are the title. All of the prototypes logical dependency finding algorithms assumed that the title of a portion of text would be formatted directly above that text.

There are many weaknesses to the prototype. All of the weaknesses arise because one of the fundamental goals of the prototype was to extract a meaningful reading order from the PDF document model rather than achieve good document understanding. Even so, it was discovered that without document class knowledge, it was almost impossible to achieve consistent results when attempting to rebuild articles or other logical entities which were spread over one or more pages. Thus, the prototype extracted a meaningful reading order of blocks from a mainly geometric perspective rather than from a totally logical perspective.

Figure 5, “Typical output from the prototype PDF processing system,” on page 70 illustrates the graphical output obtained by processing an academic type document with the prototype system. The example document is a simple, single columned document with no headers or footers. There is a hierarchy of titles present in the document which is represented in Figure 5 by colouring the bounding boxes with unique colours: green for the first level of the hierarchy; blue for the second level and yellow for the third (not shown in Figure 5).

Three types of relationships are modelled in the system: primary, secondary and tertiary. The primary relationships exist between geometric blocks of text which are tagged as main text blocks and title blocks. The prototype system displays these relationships as thick black lines to and from the centres of the appropriate blocks. Secondary relationships are displayed as slightly thinner black lines. Secondary relationships exist between title blocks. Tertiary relationships are not real logical relationships at all, but an attempt by the prototype to extract a meaningful reading order from those blocks which are not linked to by either primary or secondary relationships. The prototype uses a weak heuristic which links blocks together in the order in which it finds them on the page. The prototype searches from the top of the page to the bottom of the page, whilst traversing the blocks on the page from left to right. This algorithm is satisfactory for finding the reading order of blocks on a single page, yet, the prototype system did not attempt to find cross page links (such as those found in a newspaper, for example) which are an important feature of many documents.

## 4.4 Summary

The literary review and the development of the PDF prototype ‘understander’ have shown that the most important aspect of document understanding is obtaining the knowledge of the logical class of the document being processed. This fact is illustrated by the need, in the PDF prototype ‘understander’, to ask the user to provide the

## Document Understanding

---

document class information before choosing the appropriate technique with which to establish the logical relationships within the document.

Of all the document understanding systems reviewed only Lam's attempted any type of document classification. Lam's classification system identified categories of document which had characteristic geometric layouts and vastly different logical models. Thanks to the significant differences between any two document's geometric models, Lam could afford to adopt a model-driven classification strategy in which a document was identified by seeking out and recognising geometric features (which represent logical entities) which could only be present in one class of document.

There are some classes of document for which the differences between the geometric models of the two classes are subtle, for example, a brochure and a newsletter. Both documents may well be multi-columned, have several images per page and be formatted with a large number of fonts. Using a model-driven classification strategy on these documents would be inefficient thanks to the lack of distinguishable geometric (or logical) features which are unique to one particular class. An alternative document classification strategy must be devised which is applicable to documents which have geometric models which contain only subtle differences between them.

**The experiences and results of the prototype PDF document understanding system have preceded the final system implementation details and algorithms which are detailed in the next chapter. It was felt a review of the results of the prototype system was within the context of the chapter on document understanding, as this would illustrate the problem of document classification from the author's practical experience and from the contemporary research in document understanding. The results from the prototype have introduced the possibility of extracting fundamental logical document characteristics from a PDF document. This is a key aspect of the final system.**



# **Chapter 5, Final System Development and Engineering**

The chapter opens with a precise definition of the problem that this thesis addresses and continues with a brief description of the system development environment. The design and development of the object-oriented blackboard architecture, which forms the underlying framework of the system, is then provided. This chapter includes a detailed description of the document analysis text block segmentation routines used by the final system. A new stage of document processing is introduced which incorporates aspects of document analysis and understanding in order to extract meaningful features from a document which could help classify that document. This level of processing is implemented within the structure of the blackboard framework.

## 5.1 A precise definition of the task

All logical documents can be represented as a tree [ISO86, ISO89, Tang91]. The ‘leaves’ of the tree are specific document instances. Nodes in the tree structure represent abstract classes of logical documents. Figure 7 shows *part* of this tree.

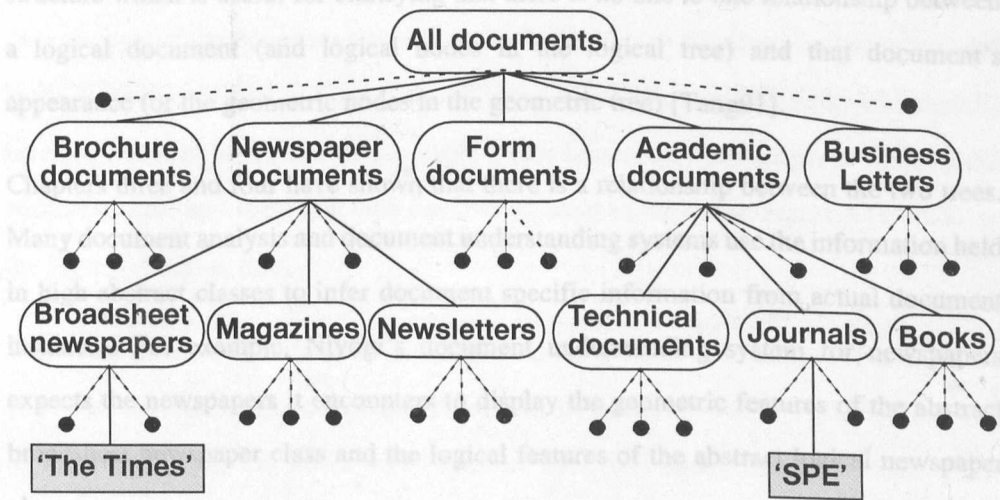


Figure 7: Part of the tree of logical documents

The tree shown in Figure 7 is not comprehensive but is designed to show that the tree of all logical documents can be structured hierarchically. There exist different levels of abstraction for different classes of document. In Figure 7 only newspaper documents and academic documents have a secondary layer of nodes beneath them. These nodes represent more detailed (and thus less abstract) classes of document. It is possible to trace ‘The Times’ (a leaf in the tree) back up the tree and see that it is a broadsheet newspaper, which is a newspaper. Thus, ‘The Times’ inherits all the logical properties of these classes of documents. ‘SPE’, on the other hand, is a journal which is an academic document.

Just as all documents can be organised into a hierarchy tree based on their logical content, so all documents can be organised into a hierarchy tree based on their geometric

properties. Geometric attributes of documents are so diverse that the corresponding hierarchy tree would contain many examples of multiple inheritance. There is no one way of formatting a logical class of documents, in fact there are a limitless number of ways of formatting a given document. The geometric tree would be extremely diverse as all the æsthetic and artistic choices of document designers and authors would have to be modelled. Consequently, the geometric tree should be considered as a hypothetical structure which is useful for clarifying that there is no one to one relationship between a logical document (and logical nodes in the logical tree) and that document's appearance (or the geometric nodes in the geometric tree) [Tang91].

Chapters three and four have shown that there is a relationship between the two trees. Many document analysis and document understanding systems use the information held in high abstract classes to infer document specific information from actual document instances. For example, Niyogi's document understanding system for newspapers expects the newspapers it encounters to display the geometric features of the abstract broadsheet newspaper class and the logical features of the abstract logical newspaper class.

How geometric document information relates to logical document information is document class specific. This was illustrated by the need to implement two different structure finding algorithms in the PDF prototype document processor. Many document understanding systems are hardcoded with the knowledge of how the geometric structure relates to the logical structure of the class of documents which they process [Ishit95, Watan93].

There are basic geometric and basic logical attributes which the majority of documents have. An example of a basic logical attribute is the presence of a main text body element or a document title element. An example of a basic geometric attribute is the number of pages that a document contains. These attributes can be raised through the abstraction hierarchy to the appropriate node of the appropriate tree. Common logical attributes are stored in the logical tree root and common geometric attributes are stored in the geometric tree root.

The level of detail of these document attributes is entirely dependent on their position in the tree. Deeper into both trees the attributes become more detailed and less abstract. At the leaves, the attributes become extremely specific, for example, the ‘main text body’ of a business letter would be logically reclassified as a ‘letter body’. A detailed geometric attribute for a certain class of documents (for example business letters) would be that they always have the senders address formatted with a right justification.

There is a relationship between certain nodes in the logical document and certain nodes in the geometric document. This is known because as humans we can make assumptions between nodes in both trees. We can take a broadsheet daily newspaper (a ‘leaf’ in the geometric tree) and identify it with the abstract class of logical newspapers (a ‘node’ in the logical structure tree).

The exact nature of the relationships between the two trees is unclear. There are no concrete rules for relating a logical document to the manner in which it is formatted. The *ad hoc* document styles, which we are familiar with, have been built up over many years and are intended to help us recognise the structure of what we are reading and to help us navigate through the document [Dillo93]. The first part of cognitive recognition is the identification of the class of document we are reading.

### 5.1.1 A formal hypothesis

Ideally, a document classification system should take the leaves of the geometric tree (which represent geometric documents) and classify them as leaves of the logical tree (which represent purely logical documents).

In fact this is an unrealistic target for several reasons. Firstly, there are thousands of ‘leaves’ on both trees. Secondly, the only guaranteed way of identifying a logical document is to search explicitly for the logical entities that it contains. This would be equivalent to taking a document and executing all known document understanding systems on it and finding the best set of results from the system’s output. Essentially,

this is what Stephen Lam's solution to document classification is: model-driven depth-first search [Lam94a]. This is workable with a few, clearly identifiable, document models to process. This technique becomes inefficient with many different document models and diminishing differences between those document models.

A better solution would be to sacrifice the level of detail of logical recognition and attempt to match nodes from the geometric tree to nodes in the logical tree. The search for the specific 'leaf' document can be resumed using a logical node in the tree as a starting point rather than the logical document root. This significantly reduces the search space, for example, a geometric document could be taken and its geometric attributes abstracted into some features which accurately and abstractly describe that documents layout and general 'feel': then logical entities could be searched for in that document. The logical entities would not be specific but instead hierarchically abstracted logical components, for example, newspapers contain logical article-title elements and journals contain section-title elements. These features can be abstracted to simply 'title' elements and this attribute raised to the class of all logical documents. By abstracting the information up the tree hierarchy the specific document instances have been replaced by abstract document classes. Extracting abstract geometric information and abstract logical information from a geometric leaf should be enough to classify that leaf into one of the abstract logical document classes.

This hypothesis can be stated formally as:

**given a geometric document, by extracting and using the abstract geometric and abstract logical properties of that document it is possible to identify that geometric document as a member of an abstract logical document class.**

There are many questions surrounding this hypothesis. How abstract do the extracted document properties (geometric and logical) have to be? How abstract does the abstract logical target class have to be? Are there any classes of document for which this hypothesis is not true?

The remainder of this chapter is devoted to describing the design and development of a system whose primary aim is to research the area defined by this hypothesis. Later chapters describe the results of the system and draw conclusions about the validity of the hypothesis based on the results produced by the final system.

## 5.2 System details

The system was developed on a PC running Microsoft Windows™ 3.11, and later Windows95™, system software. The implementation environment was Microsoft Visual C++™ version 1.5. The PC platform was chosen because as the final system was being developed and tested, the Acrobat API was available on PCs and Macintosh platforms only; the PC was the preferred development platform. The Acrobat Exchange API is implemented in C. Thanks to a degree of compatibility between C and C++, plug-in developers can choose either of these two languages to implement their plug-in with. C++ was chosen because it supports the object-oriented (OO) paradigm of programming; the author has more experience programming with C++ in an OO style than with C.

The acronym STASIS will be used to describe the final system. STASIS stands for 'System To Add Structure by InferenceS'. Throughout this chapter and the next, various figures will be used to help explain and discuss various issues. The majority of the figures are unedited (apart from cropping) screen shots of actual STASIS output.

Figure 8 shows the interface to the STASIS system together with a typical PDF file. The system was developed as a 'plug-in' to the Acrobat Exchange PDF viewer. A 'plug-in' is a dynamically linked library from which the Exchange program can call functions upon request

## 5.2.1 The STASIS interface

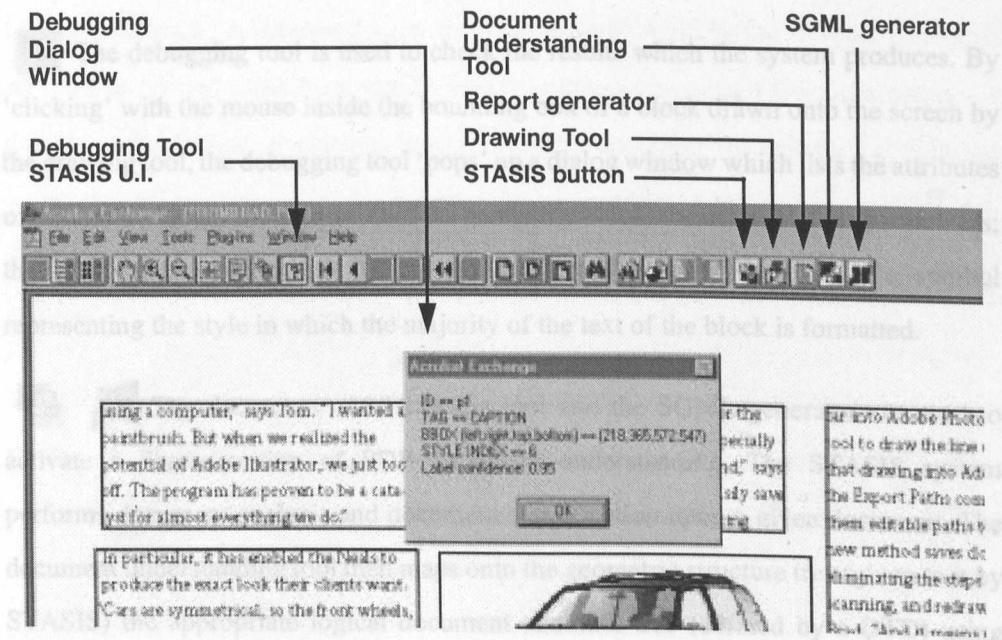




Figure 8: The Acrobat Exchange U.I. with the STASIS U.I.

A number of different tools make up the STASIS system apart from the main document classifying system. Figure 8 shows these tools in their position in the Acrobat interface. In the following paragraphs each button icon is shown next to the description of its functionality.

 The STASIS button activates the system. STASIS performs page decomposition, geometric block tagging, logical relationship finding and document feature extraction in that order before passing a feature vector to a neural net classifier for document classification.

 The report generator tool opens a report file which STASIS appends with document information each time STASIS is executed. The report contains a breakdown of all the features which STASIS has inferred or calculated from the current document.



The drawing tool, once activated, recalls the geometric blocks present on the visible page of the document and draws their bounding boxes onto the screen. Any links which are present between the blocks are also drawn.



The debugging tool is used to check the results which the system produces. By 'clicking' with the mouse inside the bounding box of a block drawn onto the screen by the drawing tool, the debugging tool 'pops' up a dialog window which lists the attributes of the chosen block: the unique ID of the geometric block; the bounding box parameters; the label of the block and the style identifier. The style identifier is a symbol representing the style in which the majority of the text of the block is formatted.



The document understanding tool and the SGML generator are used to activate a demonstration of PDF document understanding. The STASIS system performs document analysis and document classification upon a given document. The document understanding tool then maps onto the geometric structure tree (given to it by STASIS) the appropriate logical document structure tree (defined by a DTD) using document understanding routines. The SGML generator tool can then write a SGML file to disk by extracting the text which lies inside the logical blocks and applying appropriate logical tags.

## 5.3 Final System Document Analysis of PDF

STASIS uses a blackboard architecture to label the text blocks segmented from the PDF. The blackboard system has blocks provided to it by a 'Block Knowledge Source'. The role of the Block Knowledge Source is to create, store and recall the geometric blocks of the document, upon demand by the blackboard architecture controller routines (see section 5.4.1 for a description of blackboard frameworks). The creation of the geometric blocks is equivalent to the traditional definition of document image analysis in that geometric blocks are created and then tagged as being one of three classes: text, image or graphic. This section reviews the text block segmentation strategies employed by STASIS.



The test segmentation strategy which STASIS uses differs from the prototype's bottom-up strategy (see section 3.3.3, "Basic Document Analysis of PDF" for details) in that it is *hybrid*. Whereas the prototype built blocks from lines and then merged fractured blocks together, STASIS uses the text lines (given to STASIS by the Acrobat Exchange API which created them from words, which in turn were created from characters – a data-driven technique) to create columns (implicitly assuming the existence of columns – a model-driven strategy) which are then decomposed into blocks. The use of model-driven and data-driven techniques in the same system is the definition of a hybrid strategy.

The block forming routines segment text blocks a page at a time. STASIS extracts all the lines from the PDF page by using an Acrobat Exchange API function call. The lines are given attributes stating their position on the page and the font and point size of their text content. The lines are then sorted into decreasing vertical order.

### 5.3.1 Investigating PDF graphics

Research was conducted into using graphic lines on a page to help form valid text lines. Knowledge of graphic lines and their role in forming text blocks has been used in many document analysis systems [Saito93, Watan93].

The text lines provided by the Acrobat API are not always valid. It was hoped that the STASIS system could use graphic lines to validate the text lines passed to it by the API. Each text line could be checked to make sure that it did not intersect a vertical graphic line. If it did, the text line could be segmented in two at the point at which the intersection occurred. However, the generation of the graphic lines was too difficult to achieve and in practice this technique was not used.

It may seem strange that the graphic lines could not be identified easily in the PDF model. They are, after all, easily extracted and analysed from the document model. However, the identification of the valid graphic lines is almost impossible in the PDF

model thanks to PDF's ability to layer and clip graphic operators. Simply extracting all the vertical graphic lines on a page of a PDF document is not a guarantee that all the graphic lines that one has extracted are visible. Many are clipped behind other graphics including text and images. Consequently, without more detailed image processing techniques, it is impossible to tell if a graphic line is visible by the document reader.

Using the graphics regardless of their visibility produces irregular and unstable results. Text lines which do not appear to be formatted close to graphic lines become segmented. Of course, this situation does not arise in the segmentation of document images. This is one aspect of document image analysis in which analysing PDF is not advantageous.

### 5.3.2 Processing text lines

The text lines provided by the Acrobat API are analysed one by one and separated into different lists based on their font and point size attributes. A unique list of lines is generated for each unique font and point size combination. This step is based on the hypothesis that geometric blocks consist of lines of text which have the same style. The lists of lines are themselves placed into a list. For clarification the list of lists will be referred to as the meta-list and a list of lines will be referred to as a block.

Each block in the meta-list is then subjected to further text-segmentation algorithms. These algorithms were developed from Hirayama's work on column separation techniques [Hiray93] (see "Hirayama's hybrid column segmentation technique" on page 48) and the experiences gained from the prototype PDF analyser (see section 3.3.3, "Basic Document Analysis of PDF").

The system required an algorithm which could segment a document's page which has text formatted in either a multi-column style or a single column style. Hirayama's algorithm initially analyses and segments blocks using the height-distance relationships between character strings and then proceeds to segment the results by analysing the border lines of the character strings. Whilst developing the STASIS system it was found

that better results were achieved in the PDF model by analysing the leading edges of blocks and **then** analysing the height-distance relationships of the blocks created from the first stage of analysis.

### 5.3.2.1 'Democracy units'

From the experiences gained by developing the prototype document processor it was predicted that a large amount of the information that the STASIS system would extract from a document would arise from the analysis of the results of pattern recognition routines. A special abstract data type was developed to help manage the results of pattern recognition routines. The ADT is called a 'democracy unit' thanks to its ability to collect and handle 'votes'. A democracy unit stores numerical information given to it and can apply a number of different statistical functions on the stored data. A democracy unit can provide the mode, median and mean values of its stored data as well as accessing the data values themselves. Democracy units were used to help segment text blocks.

The following sections explain how a single block stored in the meta-list is segmented into valid geometric blocks. The block being analysed and segmented is referred to as the 'argument block'. The argument block is always taken and removed from the top of the meta-list. The blocks created from the segmentation of the argument block are always added to the end of the meta-list.

### 5.3.3 Analysing a block's 'leading edges'

The 'leading edge' of a block refers to the justification style of the text lines in the block. The most common styles of paragraph justification are left justified, right justified, centre justified and fully justified. Figure 10, on page 125, illustrates four columns formatted in these styles which have been segmented by STASIS. If a page contains a geometric block which is formatted with no justification (for example an advertisement) then the STASIS system will 'over segment' the geometric block into lines. The

STASIS system is designed to process all classes of document and in this case the extra work required to check for no paragraph justification is not worth the effort as it occurs extremely infrequently. STASIS adopts the policy that it is better to over segment initially and attempt to construct new blocks by merging blocks (which have been formed from over segmentation) later on in the processing cycle. This policy is an extension of the bottom-up strategy of document analysis which builds up major document components by merging together smaller components.

A new democracy unit is initialised for each argument block. Each line in the argument block casts three votes to the democracy unit. One vote is cast for the horizontal coordinate of the left most point of the line, one vote is cast for the right most horizontal point of the line and one vote is cast for the horizontal value of the centre point of the line.

The working hypothesis behind the voting system is that it should be possible to segment blocks into sub-blocks representing columns by analysing the justification attributes of the lines in the argument block. In theory the most 'popular' values that the democracy unit records votes will represent the horizontal coordinates of the leading edges of the columns. A threshold value is used to find the most popular values. The threshold is calculated using the assumption that there will be at most four columns to be found in any argument block. For every column there will be three horizontal coordinate values; thus in the case of there being four columns there will be twelve possible column leading edges. The total number of votes cast is extracted from the democracy unit and magnified by a factor of ten. The magnification ensures that in situations where there is only one column present and only a few lines in the argument block that a reasonable threshold value is still arrived at. The democracy unit is then fed the threshold value and asked to return all the values which have a magnitude equal to or greater than the threshold. The results of the threshold function are used as the parameters for the segmentation of the argument block.

The segmentation system was designed using the premise that there would be a maximum of four text columns on a page. This premise affects the assignment of the threshold value and is false for most newspapers. However, whilst testing the system it was discovered that there were no adverse effects encountered whilst segmenting pages which contained more than four columns. In practice this algorithm is a good general segmentation strategy for a page of text which has been formatted with anything up to ten columns, yet in theory this algorithm was inapplicable to pages which had anything over four text columns present.

A possible explanation for the success of this algorithm is as follows. Newspapers are the only documents where text is commonly formatted in more than four columns. Newspapers are a class of document in which the page format is extremely complex. That is to say the flow of text is frequently broken up with captions, images, sub headings, quotes and so on. The algorithm which segments a text block based on the justification style of the paragraph is designed to accommodate the worst case scenario: that each column will have three 'leading edge' values which accumulate enough votes to reach the threshold set by the segmentation algorithm. This worst case scenario would occur if there were five or more columns on a page which were all formatted in the fully justified style. The fully justified style is the only text format style which could accumulate an equal number of votes for the left, right and centre coordinates of the lines which made up its composition. Newspapers are the only class of documents in which five text columns is a realistic possibility, but thanks to the complexity of the layouts of newspapers the chance of five text columns being formatted in a fully justified style with no interference from images, captions, quotes and advertisements is very small.

### **5.3.3.1 Segmenting a block using 'leading edge' values**

For each value in the set of leading edge values found by the democracy unit, a new block is created and associated with that value. Each new block will be filled with lines which contain a match with the leading edge value associated with that block. For each line in the argument block the left most, right most and centre horizontal coordinate

values are extracted. Each value searches the set of leading edge values; if a value finds a match within the list of leading edge values the line is added to the block which has been associated with that leading edge value.

In some instances a line will have more than one horizontal coordinate value which matches up with a leading edge value. In such a case the following priorities determine which new block the line is placed into;

- **if** there is a match with the left most coordinate value of the line, **then** the line is added to the block associated with the matched leading edge value;
- **else if** there is a match with the centre coordinate value of the line, **then** the line is added to the block associated with the matched leading edge value;
- **else if** there is a match with the right most coordinate value of the line, **then** the line is added to the block associated with the matched leading edge value;
- **if** no match is made, **then** the line is added to a new block in which it is the only line present.

After all the lines in the argument block have been processed the argument block is removed from the meta-list. All the newly created blocks are added to end of the meta-list.

### 5.3.4 Analysing a block's inter-line spacing

Segmentation based on the vertical spacing between the constituent lines of a block only takes place after all the blocks in the meta-list represent the results of segmentation based on leading edge values. The working hypothesis is that the leading edge segmentation routine will have produced blocks which can be thought of as columns. These columns need to be segmented into their component geometric blocks.

The argument block to this segmentation algorithm is taken from the top of the meta-list. The results of its segmentation are added to the end of the meta-list. A new democracy unit is created for each block being analysed.

For an argument block in which the number of lines is greater than or equal to four, an inter-line gap value is calculated for every consecutive pair of lines. This value is submitted to the democracy unit. The democracy unit calculates the single most popular value. This value is referred to as the inter-line value. The argument block is then analysed using the inter-line value as a guideline for the expected leading value within the block.

For blocks in which there are less than four lines present, the following rules are applied:

- for an argument block in which there are only three text lines present two inter-line gap values can be calculated. The smallest value is chosen as the inter-line value;
- for an argument block in which there are only two text lines present the inter-line value is calculated as being the point size of the first line in the block plus 5 points. 5 points were added to the inter-line value as a margin of error 'buffer';
- for an argument block with only one line present no segmentation takes place.

### **5.3.4.1 Segmenting a block using inter-line spacing values**

At the start of the routine a new block is created. The new block contains no text lines and will be referred to as the 'current block'. The original block will be referred to as the 'argument block'.

The lines of the argument block are processed one by one. The first line in the argument block is taken from the argument block, labelled as 'current line' and added to the new block. The distance between the current line and the next line in the argument block is calculated. If the distance matches the supplied inter-line value, then the next line is

extracted from the argument block, added to the end of the current block's list of lines and becomes the current line. If the calculated inter-line gap does not match the supplied inter-line gap, then the current block is sealed and added to the end of the meta-list; a new block is created, it becomes the current block and the first line in the argument block's list of lines becomes the current line. The current line is added to the current block and the algorithm continues. Figure 9 shows a flow chart which illustrates this algorithm.

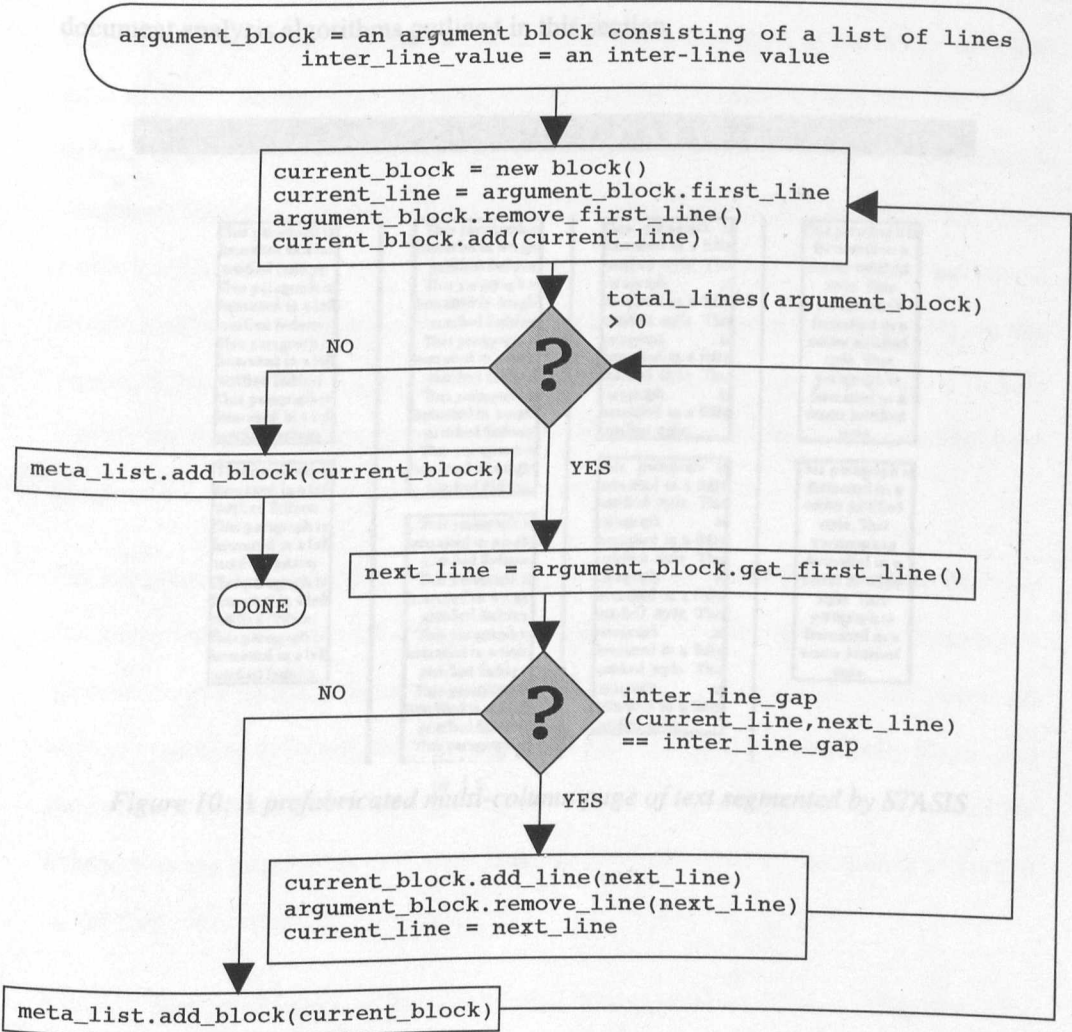


Figure 9: A flow chart of the segmentation routine based on inter-line gap values



Figure 10, on page 125, shows a page of text which has been formatted to provide a demonstration of the capabilities of the text block segmentation algorithms outlined in this chapter. Figure 10 shows, from left to right, a left justified text column, a right justified text column, a fully justified text column and a centrally justified text column. The text columns have also been broken into logical paragraphs by using blank lines. This example is unusual in that typical multi-column page formats are not so simple, and they do not provide such well defined columns. The next chapter (“Results”) defines and provides examples of error forming contexts and valid block forming contexts for the document analysis algorithms outlined in this section.

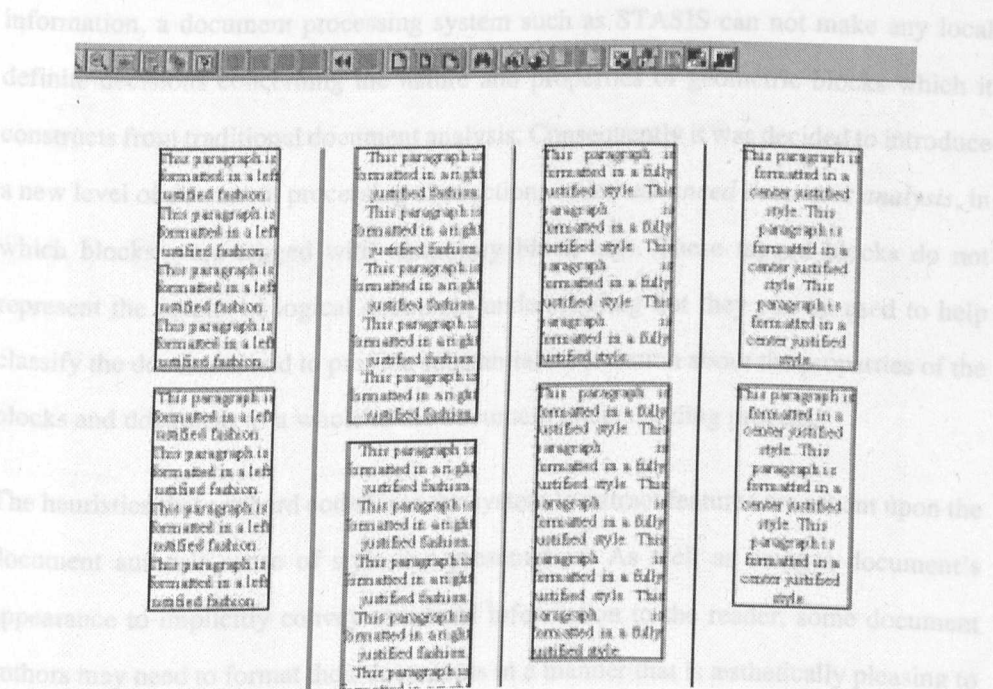


Figure 10: A prefabricated multi-column page of text segmented by STASIS

## 5.4 Development and design of the blackboard architecture

From the experience gained through the Adobe prototype implementation it was deduced that the core of the non *a priori* document analysis and understanding problem was that certain specific document understanding algorithms can only be applied once the class of document being processed is known. This also applies in a limited fashion to document analysis. If the analysis system has knowledge of the class of document it is analysing then it can perform a top-down analysis. Deprived of the document class information, a document processing system such as STASIS can not make any local definite decisions concerning the nature and properties of geometric blocks which it constructs from traditional document analysis. Consequently it was decided to introduce a new level of document processing abstraction; called *advanced document analysis*, in which blocks were tagged with temporary block tags. These tagged blocks do not represent the results of logical document understanding but they can be used to help classify the document and to provide substantial information about the properties of the blocks and document as a whole to the document understanding process.

The heuristics that are hard coded into the system to extract features are reliant upon the document author's sense of style and presentation. As well as using a document's appearance to implicitly convey structural information to the reader, some document authors may need to format their documents in a manner that is aesthetically pleasing to the eye. This typically occurs for product brochures and presentation slides, less so for newspapers and magazines, to an even lesser degree for academic documents and not at all for form documents.

STASIS needed to have a framework which supports artificial intelligence type reasoning and the handling of uncertain data in order to process a potentially limitless set of documents, each of which could be formatted in a unique style. STASIS needed to be able to sieve through the precise details of a document's design and extract general

features based upon inferences made by knowledge sources which have knowledge about *de facto* document formatting rules hard coded into their knowledge bases.

A blackboard architecture was decided upon because it provided the framework to model the document classification problem in terms of abstract hierarchical layers. These layers could be built up by individual expert knowledge sources co-operating to solve sub goals with a variety of problem solving techniques. Furthermore, blackboard systems are opportunistic: they react in an appropriate fashion when the appropriate opportunity arises. This suits a universal document processing system in which different classes of document are to be processed. The different documents will present the system with wide ranging problems which occur in different orders from document to document. The system architecture needs to be flexible enough to handle the degree of variance present in document layouts.

### 5.4.1 Blackboard systems

There follows a brief introduction to the blackboard framework which is intended to describe the fundamental principles and theories behind a blackboard system.

In an opportunistic reasoning model, pieces of knowledge are put forward, or retracted, at the most opportune time. The blackboard model of problem solving is a highly structured, special case of opportunistic problem solving. As well as applying opportunistic reasoning as a knowledge-application strategy, the blackboard model prescribes the organization of domain knowledge and all the intermediate and partial solutions needed to solve the problem at hand. Englemore, Morgan and Nii describe the blackboard problem solving model using the metaphor of solving a jigsaw puzzle.

*"Imagine a room with a large blackboard and around it a group of people each holding oversize jigsaw peices. We start with volunteers who put on the blackboard (assume it's sticky) their most 'promising' pieces. Each member of the group looks at his pieces*

*and sees if any of them fit into the pieces already on the blackboard. Those with the appropriate pieces go up to the blackboard and update the evolving solution. The new updates cause other pieces to fall into place, and other people go to the blackboard to add their pieces. It does not matter whether one person holds more pieces than another. The whole puzzle can be solved in complete silence; that is, there need be no direct communication among the group. Each person is self activating, knowing when his pieces will contribute to the solution. No a priori established order exists for people to go up to the blackboard. The apparent cooperative behaviour is mediated by the state of the solution on the blackboard. If one watches the task being performed, the solution is built incrementally (one piece at a time) and opportunistically (as an opportunity for adding a piece arises), as opposed to starting systematically from the top left corner and trying each piece.” Engelmores [Engel88]*

The solution space is organized into application dependent hierarchies. The hierarchies may be an abstraction hierarchy, a part-of hierarchy or any other type of hierarchy appropriate for solving the problem. A similar flexibility is shown towards the choice of inference methods available. At any stage of the processing either forward, backward, event driven, goal driven, expectation driven or data driven reasoning may be applied.

Figure 11 illustrates that the blackboard framework consists of three elements: a blackboard, multiple knowledge sources and a controller that mediates amongst the knowledge sources. The hierarchies present on the blackboard are shown in Figure 11 as object relationships. Figure 11 is an adaptation of Booch's blackboard framework diagram [Booch91].

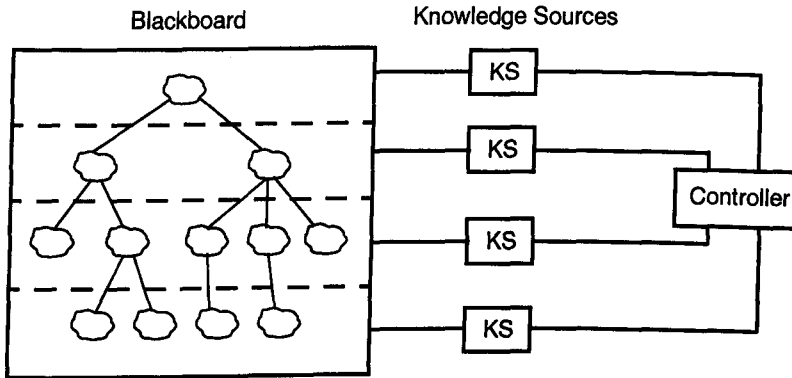


Figure 11: A blackboard framework

### 5.4.1.1 Knowledge sources

The knowledge required to solve the problem is partitioned into knowledge sources that are kept separate and independent. The objective of each knowledge source is to contribute information that will lead to the solution of the problem. Each knowledge source is responsible for knowing the conditions under which it can contribute to a solution; each knowledge source has pre-conditions that indicate the condition on the blackboard that must exist before the knowledge source is activated. The knowledge sources can only modify the blackboard or control data structures which lie on the blackboard.

### 5.4.1.2 Blackboard data structures

The blackboard can be thought of as a global database. Interaction between knowledge sources takes place solely through changes to the blackboard. The blackboard holds computational and solution state data needed by and produced by the knowledge sources. Typically, blackboard objects include input data, partial solutions, alternative solutions, final solutions and control data. The objects are hierarchically organized into levels of analysis. Information associated with objects on one level serves as input to a set of knowledge sources which, in turn, place new information on the same or other levels [Engel88].

### 5.4.1.3 The Controller

Control routines monitor the changes which take place on the blackboard and decide what actions to take next. Various kinds of information are made available to the control routines. The control routines establish where the 'focus of attention' of the system is. The focus can either be the knowledge sources (i.e. which knowledge sources to activate next); blackboard objects (i.e. which solution goals to pursue next), or a combination of both (i.e. which knowledge sources to apply to which blackboard objects). The solution is built one step at a time. Any type of reasoning can be applied at each stage of the solution formation. Thus the sequence of knowledge source invocation is dynamic and opportunistic rather than fixed and preprogrammed [Engel88].

### 5.4.1.4 Problem solving behaviour and knowledge application

The problem solving behaviour is determined by the knowledge-application strategy of the control modules. The choice of the strategy is dependent on the characteristics of the application task and on the quality and quantity of the domain knowledge relevant to the task. A commitment to a particular reasoning process is a commitment to a particular knowledge-application strategy. STASIS is committed to data-driven problem solving as it is trying to find the logical class of a document by analysing large geometric document components which have been constructed by studying smaller geometric components. Therefore, the knowledge application strategy that the STASIS system should adopt dictates that a knowledge source should be selected which only places data on the blackboard on an equivalent hierarchical level or a greater level.

### 5.4.1.5 Example blackboard systems

Two blackboard systems which solve image processing problems solutions using two different knowledge application strategies are presented. A data-driven blackboard based scheme was developed by Draper et al. [Drape88] for understanding images or pictures. Nagao et al. [Nagao88] have written a blackboard system which structurally

analyses complex aerial photographs from a model-driven approach. Both systems provide relevant information to the development of the STASIS system and are worthy of discussion.

Draper's system attempts the enormous task of recognising objects in everyday photographs. He has developed an extension to the normal blackboard problem solving paradigm which uses schemas to help split up the global problem. Schemas can be thought of as knowledge sources which are 'intelligent' and 'autonomous'. The centralised control techniques which are typical of most blackboard systems are rejected in favour of running parallel schemas. Knowledge sources still exist in this system but their role in the system is slightly different due to the increased power the schemas have in directing their own reasoning processes.

The manner in which Draper handles uncertainty is extremely relevant to the STASIS system. Draper acknowledges that local errors are created by imperfect knowledge sources (he states segmentation and line grouping processes as examples) and incorrect heuristics. Draper incorporates numerical and symbolic uncertainty reasoning techniques in his system. Numerical techniques include Bayesian probability theory [Stutz94], fuzzy set theory [Zadeh83] and the Shafer-Dempster theory of evidence [Jacks92, Shafe76]. Draper claims that numerical techniques may suffer from a loss of information in the domain of artificial intelligence. Symbolic approaches, on the other hand,

*"explicitly record the sources of uncertainty in order to facilitate situation-specific combination methods and recovery from errors."*

Draper [Drape88].

One of the possible problems with the symbolic approach is that it is so flexible that analysing every situation-specific error combination could be potentially inefficient for a system. Draper issues a reassurance claiming that only a few combinations will be relevant and that using numerical and symbolic techniques together produces good results.

Nagao's system [Nagao88] attempts to locate houses, forests, crop fields, bare soil, roads and cars from the analysis of aerial photographs. The advantage that Nagao's system gained from being implemented with a model-driven knowledge application strategy is that the properties of already recognised objects can be used in order to analyse unrecognised areas and to find context-dependent objects. It should be noted that Nagao's system searches for certain well defined shapes in the image. Consequently, it can benefit from a model-driven search technique which is traditionally more accurate when presented with suitable data.

### 5.4.2 Object-oriented system design

The basic principles of designing a blackboard system in an object-oriented fashion are outlined by Booch in a chapter devoted to solving a 'cryptanalysis' problem [Booch91]. The underlying structure of a blackboard system does not change from problem domain to problem domain; thus the following sections are based on Booch's design processes with regards to developing a blackboard system. The biggest difference between the system developed by Booch and the STASIS system is the handling of dependencies, assumptions and inference engines by the knowledge sources. The STASIS system is solving a similar problem every time it is activated. Consequently, there is no need to encode a sophisticated inter-knowledge source assumption and dependency protocol as the knowledge sources will each perform a well defined function at various stages of the system execution regardless of the document being processed. Booch's example is a general blackboard framework which is designed to help the reader's understanding of the principles of object-oriented programming rather than the principles of a working blackboard architecture.



5.4.2.1 Analysis of the knowledge sources

The requirements of the knowledge sources have been built up from a variety of sources. The most influential source was the PDF document processing prototype. The development of the prototype helped to identify and encapsulate the knowledge sources which infer knowledge from the document. Other knowledge sources were deemed relevant by analysing the cognitive processes which help us classify documents, for example, looking at the number of columns, or the distribution of images throughout the document. The creation of the majority of the knowledge sources was done during the design process. However, thanks to the ‘reusability’ of the object-oriented paradigm coupled with the abstract and encapsulated nature of the role of knowledge sources within the blackboard framework, new knowledge sources were designed, implemented and added throughout the evolution of the system. Currently, the STASIS system uses eleven knowledge sources. They are listed together with the knowledge they ‘encapsulate’ in Table 12.

| Knowledge Source | Encapsulated Knowledge  |
|------------------|---|
| Text Block       | The geometric parameters of all the geometric blocks in the document. |
| Text Frequency   | Which text blocks make up the body of the document.                   |
| Super Title      | Which text block(s) is the document title.                            |
| Title            | Which text blocks are logical titles within the document.             |
| Image            | Which blocks are image blocks.  |
| Footer           | Which text blocks are footer blocks.                                  |
| Header           | Which blocks are header blocks.                                       |
| Graphic          | Knowledge of the straight graphical lines in the document.            |
| Caption          | Which text blocks are image captions.                                 |
| Column           | The parameters of the text columns on all the pages.                  |
| Document Class   | The class of document being processed.                                |

Table 12: STASIS knowledge sources and the knowledge they ‘encapsulate’

Each knowledge source represents a possible class in the object-oriented system implementation. Booch declares that

*“...each knowledge source embodies some state, each exhibits certain class-specific behaviour and each is uniquely identifiable.”*

Booch [Booch91]

5.4.2.2 Design of the blackboard

The STASIS blackboard is a simple global data storage object. Only blackboard objects can be placed on the blackboard or removed from it. The blackboard objects are used to contain information which is useful in achieving the goal or sub-goals of the system. The STASIS system has seven blackboard objects which are listed with their role and functionality in Table 13.

| Blackboard Object     | Role and Functionality  |
|-----------------------|---|
| List of blocks        | A list object which contains geometric blocks which are examined by the knowledge sources.  |
| Page Number           | The number of the page which is being processed.  |
| Total Number of Pages | (self explanatory).   |
| Utility List          | A miscellaneous data structure which can be used by knowledge sources to keep partial results.  |
| System State Marker   | This is a flag which the controller can use to pass information to various elements in the system regarding the current state of the system.                                |
| Document Vector       | This is a binary vector value which is gradually filled in by the knowledge sources during the processing. It is passed to a neural classifier for document classification. |
| Document Identifier   | The is a variable which stores the result of the neural net classification process.   |

Table 13: STASIS blackboard objects

Figure 12 shows an inheritance diagram which illustrates the relationship between the blackboard object base class and the classes which inherit from it. Figure 12 indicates that there is no restriction on the number of blackboard objects which can be placed on the blackboard.

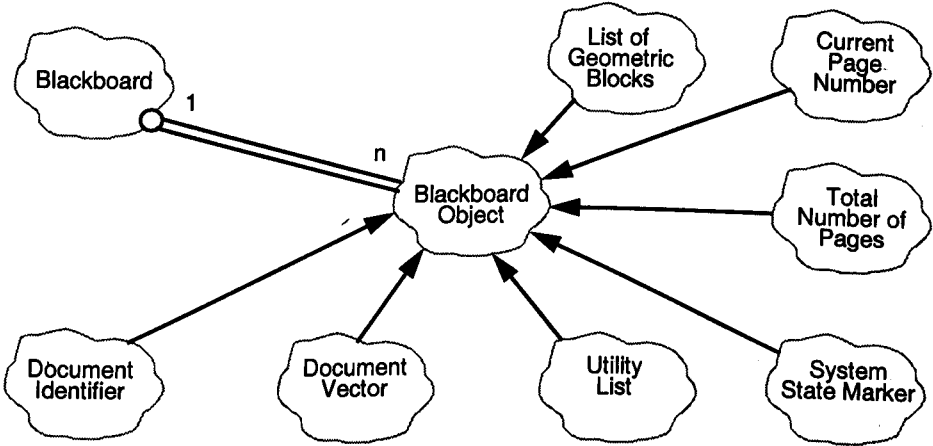


Figure 12: The blackboard and blackboard object class diagram

5.4.2.3 Design of the knowledge sources

The analysis of the knowledge sources lead to the identification of eleven separate knowledge source classes. In a similar manner to the development of the blackboard objects, a base class was developed to elevate the common features and functions of the knowledge sources. A certain subsection of the knowledge sources operate purely on text blocks. However there was not enough common functionality between these knowledge sources to justify creating another layer of abstraction. Consequently, all the knowledge sources bar three inherit directly from the knowledge source base class.

A description of the implementation of the knowledge sources' algorithms is provided in section 5.5.2. The remainder of this section outlines the knowledge sources which are virtual base classes for other knowledge sources.

The Structure Knowledge Source is used to create relationships between blocks. The STASIS system makes geometric relationships between main text blocks and logical dependency relationships between title blocks and other blocks. A geometric relationship is created between main text blocks which STASIS believes relate to one another based on their geometric positions and attributes, for example two blocks may be directly above one another, in the same column, in the same text style and with the same inter-line gap value. This much evidence would be enough for STASIS to assume that there is a relationship between these two blocks. The relationship is not logical but geometric in its nature.

A logical dependency relationship exists between a title block and the blocks under that title block's *umbrella of influence*. The title block is logically dependent on the blocks within its influence; without their presence it would not be a title. The Structure Knowledge Source 'knows' the criteria for making both these types of relationship. It is called upon to create these relationships after the geometric blocks have been labelled. The Title Knowledge Source inherits from the Structure Knowledge Source so that it may benefit from the knowledge of how to make logical relationships between title blocks and other blocks. This knowledge is useful in justifying labelling a candidate title block as a *bona fide* title block.

The Footer and Header Knowledge Sources have very similar roles and responsibilities within the system. They both search for recurring patterns amongst blocks, with which they can identify headers and footers. An abstract class entitled 'peripheral' was created to elevate the common attributes of these knowledge sources. The name peripheral refers to the position that headers and footers have in any geometric document; they are always found on the perimeter of the pages.

Figure 13 is the knowledge source class diagram which also describes the role of the 'KnowledgeSources' class as a container for the knowledge sources.

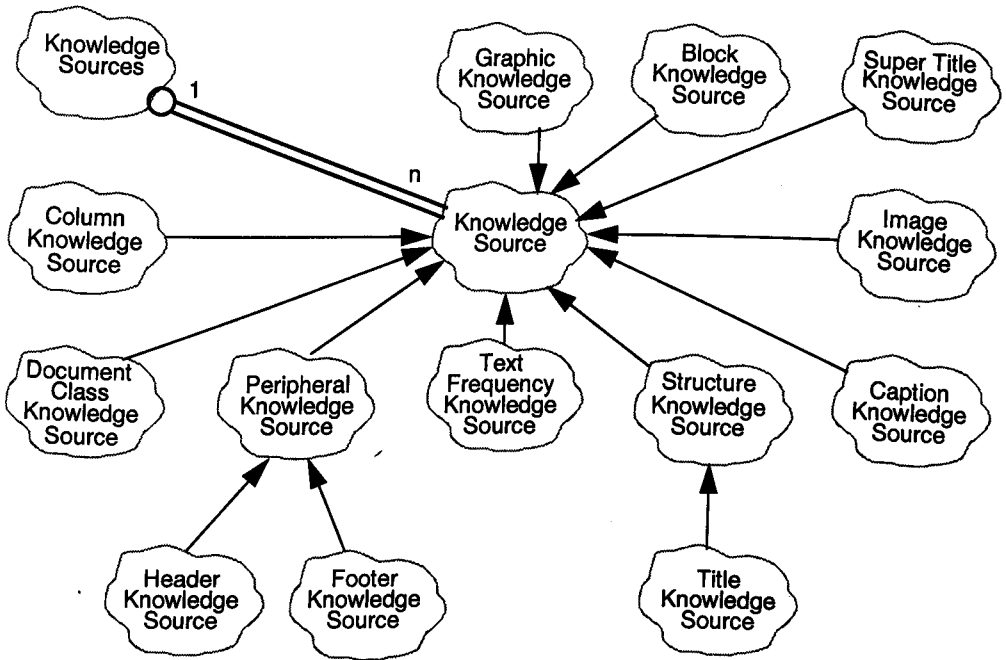


Figure 13: The STASIS knowledge source class diagram

## 5.4.2.4 Design of the controller

Figure 14 shows how the controller and the individual knowledge sources interact. The square 'F' blocks on the objects indicate a 'friend'<sup>1</sup> relationship. At any stage in the system execution cycle a particular knowledge source may decide that it has something useful to add to, or learn from the contents of the blackboard and it will give a hint to the controller. Once all the knowledge sources have been given a chance, the controller selects the most promising hint and allows the appropriate knowledge source access to the blackboard. Each hint is given a weighting by the knowledge source which owns it. The value of the weighting can be thought of as a confidence factor. The controller has a variety of methods of selecting which hint to follow which depend on the current working state of the system. These methods will be explained in the next section.

1. "A non-member function which is allowed access to the private part of a class [without requiring membership] is called a friend of the class" Stroustrup, p161 [Strou91]

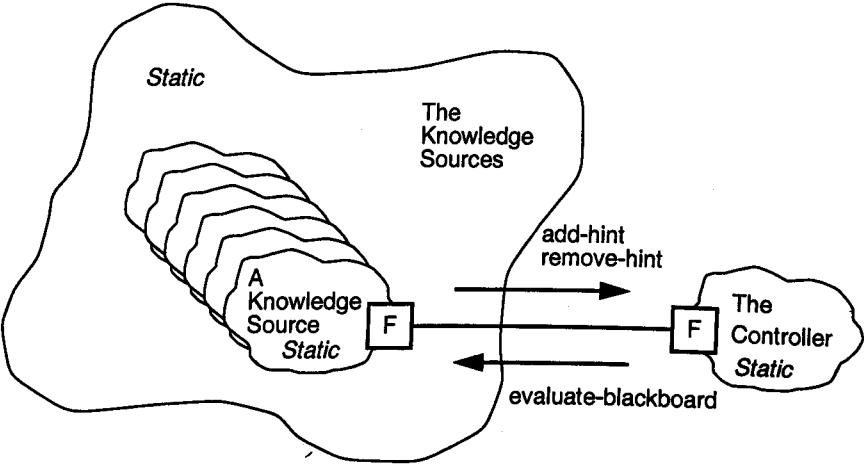


Figure 14: The controller mechanism

5.4.2.5 Tailoring the blackboard framework to the document processing problem

Figure 15 is an object diagram which illustrates the relationships between the topmost objects in the STASIS system. This diagram should be compared to the generic structure of a blackboard framework illustrated in Figure 11, on page 129.

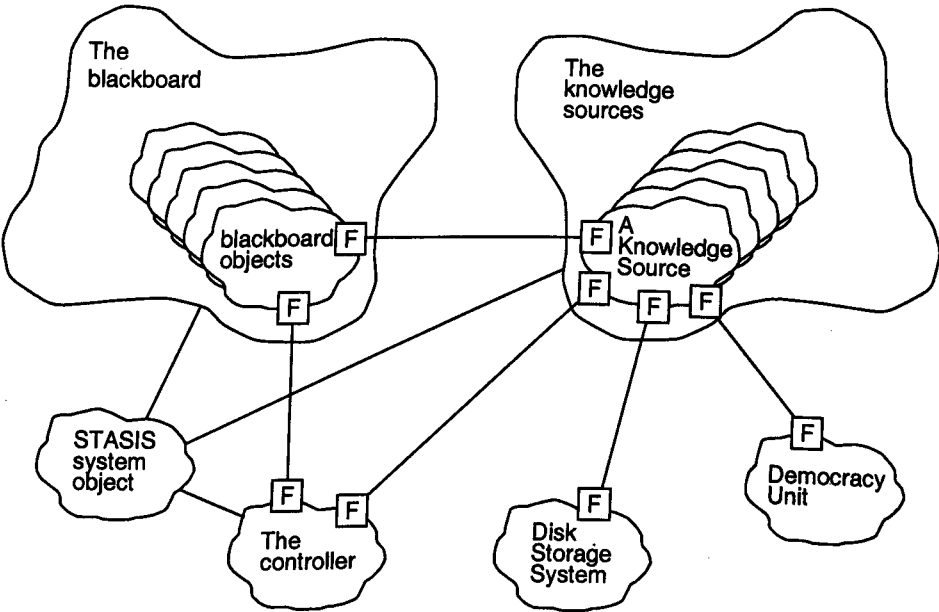


Figure 15: STASIS object diagram

Two of the classes which are present in Figure 15 are the DemocracyUnit (DU) and the DiskStorageSystem (DSS). The DU is a simple abstract data type class which keeps a tally of entities given to it by creating a histogram. The DU has various data manipulation routines which provide the user with statistical information regarding the data. A description of a working example of a DU is provided in section 5.3, "Final System Document Analysis of PDF". The DSS is another abstract data class which maintains a database of all the geometric objects which are handled by the system. The database is stored on disk. Objects are recalled either a page at a time, or individually, when required. All the knowledge sources have access to instances of both these classes.

The STASIS class encapsulates the blackboard, the knowledge sources and the controller. It is a simple class whose operations are to:

- connect the blackboard, the controller and the knowledge sources;
- restart the system;
- solve the document classification problem by activating the system.

The knowledge sources are allowed to see the blackboard and the blackboard objects. This relationship is represented by a single connection from an individual knowledge source to an individual blackboard object.

The STASIS system controller has access to both the knowledge sources and the blackboard. This is unusual for most blackboard systems but not unheard of. The controller can receive hints from the knowledge sources and change the particular blackboard object which indicates the current state of the STASIS system: the state indicator (SI).

The presence of the SI allows the STASIS system to cut out a number of routines which the architecture of the generic blackboard framework enforces. This increases the efficiency of the system at the expense of diluting the traditional blackboard

architecture. There are five processing states in the document classification strategy which STASIS adopts; all of which are clearly defined and incremental. Table 14 identifies those stages and describes the major activity of the stages.

| Processing Stage | Stage Activity   |
|------------------|--|
| 1: Train         | Create the geometric blocks and present them to the knowledge source for training. |
| 2: Label         | Tag the geometric blocks based on the knowledge embodied in the knowledge sources. |
| 3: Structure     | Find logical relationships between tagged blocks.                                  |
| 4: Infer         | Extract document features from the document.                                       |
| 5: Classify      | Classify the document by feeding the document features to a neural net.            |

**Table 14: STASIS processing stages**

Although the document classification problem can be partitioned into stages and sub goals which follow each other in an incremental order, the processes and routines which need to be performed within the stages require the flexibility of a blackboard framework.

The knowledge sources and the controller have different responsibilities during different stages of processing. By examining the State Indicator (SI) and the status of the blackboard objects the knowledge sources can decide whether or not to send a hint to the controller. STASIS uses relatively few individual blackboard objects. Consequently, without the SI it would be impossible for a knowledge source to unambiguously interpret the state of the system from the configuration of the blackboard objects.

The controller adopts a different hint selection policy at different processing stages. Frequently in stage one, more than one knowledge source sends a hint to the controller asking to be activated. In stage one (knowledge source training) every hint should be acted upon, without preference. Thus the controller simply activates all the hints, effectively granting access to all the knowledge sources who request it.



Stage two (block labelling) asks the knowledge sources to tag the geometric blocks in the document based on the embodied knowledge they carry in themselves. This knowledge is either hard coded or inferred during the training stage. Frequently, more than one knowledge source will wish to tag a block with a certain tag. However, the controller can only grant one knowledge source the right to activate itself and tag the block; 'conflict resolution' is the term used to describe the choice the controller makes in this situation.

The STASIS controller uses two techniques for conflict resolution among knowledge sources. There are two types of knowledge source conflicts: those that involve the Title Knowledge Source and those which do not.

The Title Knowledge Source is the only knowledge source which requires information about the blocks which are formatted beneath the block which it is trying to tag. This information can be thought of as contextual information. The Title Knowledge Source can assert with varying confidence that a block is a title if it has knowledge of the geometric context of that block.

All other knowledge sources in the STASIS system do not require this contextual information. In this case the controller uses a numerical method of reasoning about uncertainty. The controller can resolve conflicts between knowledge sources which do not involve the Title Knowledge Source by simply comparing the confidences of the knowledge source's assertions.

The controller resolves conflicts between knowledge sources which involve the Title Knowledge Source by using symbolic uncertainty methods. Symbolic reasoning is described by Draper [Drape88] as being effective in limited environments, for example, in contexts in which the number of possible uncertainty scenarios is small. The STASIS system has twelve knowledge sources in total.

The following equation shows a formula, derived from a series, which calculates the number of possible combinations of two or more knowledge sources.  $x$  is the number of knowledge sources.

$$x = \sum_{i=0}^{i=(x-2)} \frac{x!}{(x-i)!i!} = 2^n - (n+1)$$

When  $x$  is twelve there are 4083 possible conflicts. However, of those twelve knowledge sources only six are involved in the labelling process. When  $x$  is six there are 57 conflicts.

This is still too many possible conflict scenarios. Fortunately, one can discount the super Title Knowledge Source and the Text Frequency Knowledge Source from the problem as both these knowledge sources produce assertions which are invariably correct. When  $x$  is 4 there are 11 conflict scenarios which is much more acceptable. The four knowledge sources still left are the Title, Footer, Caption and Header Knowledge Sources. The Header and Footer Knowledge Sources are derived from the same base class (the Peripheral Knowledge Source), and it is this base class which handles the assertion of labelling confidences. Thanks to the hierarchical raising of this functionality there are effectively only three knowledge sources which produce conflicts. This generates four possible scenarios. As all these conflicts involve the Title Knowledge Source the description of their resolution will be described later together with the functionality of the Title Knowledge Source in section 5.5.2.5.

Faced with having to resolve a conflict, the controller simply selects the hint which has the highest activation weighting. The activation weighting can be seen as a confidence factor, set by the knowledge source, representing the level of confidence that the knowledge source has in its own judgement.

## Final System Development and Engineering

---

In situations in which two hints have the same confidence factor, the controller simply picks the last hint it sees with the appropriate confidence factor. This situation is avoidable by engineering the confidence factors that the knowledge sources give to their hints. For each knowledge source, one can:

- assess the effectiveness of the training strategies employed by the knowledge source;
- assess the difficulty of the sub-goal the knowledge source is attempting;
- build up the total confidence factor of the knowledge source's hint by attributing partial confidences to the positive results of certain tests which the argument block is subjected to.

Stage three (structure inferring) is a unique stage in that only the structure knowledge source's hints can be executed during this stage. The Structure Knowledge Source cycles through the document's pages, extracting all the blocks, and finding logical relationships between the blocks based on their geometric attributes and their semi-logical tags. Initially, this stage was designed to find the 'reading order' (see section 4.2.8, "Saitoh's text area ordering system" for a definition of 'reading order') amongst the blocks. However, during the system development it was decided that such a task was primarily the responsibility of a document understanding system. Consequently, this stage was not developed thoroughly.

In stage four all the knowledge sources infer document features from the knowledge they have accrued from examining the blocks and the relationships between the blocks during the previous three stages. The controller activates any knowledge source which wishes to be activated during this stage. No conflict resolution technique is required. Stages one to four are collectively termed advanced document analysis.

Stage five (document classification) is similar to stage three in that only one knowledge source sends a hint to the controller in this stage. Only the Document Class Knowledge Source requires activation, as it needs to copy the document feature vector blackboard object into itself and feed it to the neural net.

This section has provided an overview of the internal structure of the system and a high level view of the workings of the system. The next section describes in some detail the algorithms for training the knowledge sources, labelling the blocks and extracting and generating the document features.

## **5.5 Advanced document analysis: the generation of document features**

Traditional document analysis algorithms do not provide enough geometric information to clearly pinpoint the document's logical class. Detailed analysis algorithms such as Sivaramakrishnan's [Sivar95], which can state with confidence some of the attributes of the text present in the document, do not provide sufficient information to help recognise the logical class.

This research has produced an algorithm which can classify a geometric document's logical class if the geometric features in the document follow the patterns found in other documents of the same logical class. The blackboard framework subjects the geometric blocks extracted from the traditional document analysis routines to further geometric analysis. The geometric blocks are also analysed from a logical perspective. The abstraction level of the logical analysis is low; it has to be low in order to be applicable whilst the class of the document is still unknown. The blackboard framework does not perform the classification itself but provides a document feature vector to a neural net classifier.

### 5.5.1 Extraction of meaningful document features

The feature vector was continually engineered during the development of the final system. Some of its features are purely geometric and were simply extracted from the PDF document model, for example, the number of pages in the document. Newspapers and magazines typically have a large number of pages; forms typically have less than ten pages. Other features are found using basic logical analysis, for example, the number of unique title block styles within the document is a good guide to the number of different logical title levels in the document. Newspapers usually have a large number of title block styles; academic documents have relatively few title styles.

The majority of the features are extracted from the geometric analysis of the PDF document, for example the relationship between a document's images and its text columns. These features are designed to represent attributes of the document layout which would be typical of documents from the same logical class and yet have a different state in documents from a different logical class.

There are forty nine features in total. Each feature is represented as a binary value. Table 15 lists the document features and their indices.

| Index | Feature Descriptor    | Heuristic Description                            |
|-------|-----------------------|--|
| 1     | DOUBLE_SIDED          | True if document is double sided                 |
| 2     | NUM_OF_PAGES_1_TO_4   | True if 1 to 4 pages present                     |
| 3     | NUM_OF_PAGES_5_TO_9   | True if 5 to 9 pages present                     |
| 4     | NUM_OF_PAGES_10_TO_20 | True if 10 to 20 pages present                   |
| 5     | NUM_OF_PAGES_20_PLUS  | True if more than 20 pages present               |
| 6     | MAIN_TEXT_1           | True if one main text style present              |
| 7     | MAIN_TEXT_2           | True if two main text styles present             |
| 8     | MAIN_TEXT_3           | True if three main text styles present           |
| 9     | MAIN_TEXT_3_PLUS      | True if more than three main text styles present |

Table 15: Document features and their indexes.

## Final System Development and Engineering

|    |                             |   |
|----|-----------------------------|---|
| 10 | TEXT_STYLES_1_TO_4          | True if 1 to 4 total text styles present              |
| 11 | TEXT_STYLES_5_TO_9          | True if 5 to 9 total text styles present              |
| 12 | TEXT_STYLES_10_TO_19        | True if 10 to 19 total text styles present            |
| 13 | TEXT_STYLES_20_TO_40        | True if 20 to 40 total text styles present            |
| 14 | TEXT_STYLES_40_PLUS         | True if more than 40 text styles present              |
| 15 | HEADERS                     | True if headers present                               |
| 16 | HEADER_STYLE_CONSISTENCY    | True if style of headers is consistent                |
| 17 | HEADERS_25                  | True if headers present on 25% of pages or more       |
| 18 | HEADERS_50                  | True if headers present on 50% of pages or more       |
| 19 | HEADERS_75                  | True if headers present on 75% of pages or more       |
| 20 | FOOTERS                     | True if footers present                               |
| 21 | FOOTER_STYLE_CONSISTENCY    | True if style of footers is consistent                |
| 22 | FOOTERS_25                  | True if footers present on 25% of pages or more       |
| 23 | FOOTERS_50                  | True if footers present on 50% of pages or more       |
| 24 | FOOTERS_75                  | True if footers present on 75% of pages or more       |
| 25 | CAPTIONS                    | True if captions present                              |
| 26 | CONSISTENT_CAPTIONS         | True if style and position of captions is consistent  |
| 27 | CONSISTENT_CAPTION_POSITION | True if position of captions consistent               |
| 28 | CONSISTENT_CAPTION_METRIC   | True if style of captions consistent                  |
| 29 | TITLES_0_TO_3               | True if 0 to 3 titles present                         |
| 30 | TITLES_4_TO_9               | True if 4 to 9 titles present                         |
| 31 | TITLES_10_TO_20             | True if 10 to 20 titles present                       |
| 32 | TITLES_20_PLUS              | True if more than 20 titles present                   |
| 33 | COLUMNS_1                   | True if one column is maximum throughout document     |
| 34 | COLUMNS_2                   | True if two columns are maximum throughout document   |
| 35 | COLUMNS_3                   | True if three columns are maximum throughout document |

**Table 15: Document features and their indexes.**

|    |                      |  |
|----|----------------------|--|
| 36 | COLUMNS_3_PLUS       | True if more than three columns is maximum throughout document       |
| 37 | INCONSISTENT_COLUMNS | True if column style is inconsistent                                 |
| 38 | IMAGES_0             | True if no images present in document                                |
| 39 | IMAGES_5             | True if less than 0.5 images present per page                        |
| 40 | IMAGES_5_TO_9        | True if 0.5 to 0.9 images present per page                           |
| 41 | IMAGES_10_PLUS       | True if more than 1.0 images present per page                        |
| 42 | IMAGES_CONSISTENCY   | True if image styles are consistent                                  |
| 43 | IMAGES_STRADDLE      | True if some images straddle columns                                 |
| 44 | IMAGES_IN_LINE       | True if some images are in line with columns                         |
| 45 | IMAGES_NEXT_TO       | True if some images are next to columns                              |
| 46 | LINE_60              | True if average number of straight lines per page is greater than 60 |
| 47 | LINE_40              | True if average number of straight lines per page is greater than 40 |
| 48 | LINE_20              | True if average number of straight lines per page is greater than 20 |
| 49 | LINE_10              | True if average number of straight lines per page is greater than 10 |

**Table 15: Document features and their indexes.**

There follows a description of the document features plus their expected values in certain classes of document.

- (1) Feature 1 is set to true if the system detects headers or footers which it suspects may be formatted with a double sided page layout. Newspapers and magazines usually have a double sided page layout. The knowledge source responsible from inferring this feature is the Peripheral KS.
- (2-5) Features 2 to 5 are used to record the number of pages in the document. They are mutually exclusive. Feature 2 is true if there are less than 4 pages present. Feature 3 is true if there are 4 to 9 pages present. Feature 4 is true if there are 10 to 20 pages present. Feature 5 is true if there are greater than 20 pages present.

- (6-9)** Features 6 to 9 are used to record the number of main text styles detected in the document. Respectively, features 6, 7, 8 and 9 are set to true if there is 1 main text style, 2 text styles, 3 text styles or more than 3 text styles present in the document. They are mutually exclusive. Newspapers and magazines are more likely to have more than 1 main text style.
- (10-14)** Features 10 to 14 record the total number of unique text styles present in the document. Respectively, features 10, 11, 12, 13 and 14 are set to true if there is less than 5 text styles present, 5 to 9 styles, 10 to 19 styles, 20 to 40 styles or greater than 40 text styles. Newspapers and magazines have a large number of unique text styles.
- (15-19)** Features 15 to 19 describe the geometric properties of the headers present in the document. Feature entry 15 is true if any geometric blocks are tagged with the header tag. If feature entry 15 is set to false, implying no headers exist, then features 16 to 19 are set to false. Feature 16 is set to true if all the headers are formatted in the same text style. Features 17 to 19 describe the number of pages which have headers present on them. Respectively, features 17, 18 and 19 are set to true if there are headers on over 25% of all pages, over 50% of all pages and over 75% of all pages. Features 17 to 19 are mutually exclusive.
- (20-24)** Features 20 to 24 describe the geometric properties of the footers present in the document. Feature entry 20 is true if any geometric blocks are tagged with the footer tag. If feature entry 20 is set to false, implying no footers exist, then features 21 to 24 are set to false. Feature 21 is set to true if all the footers are formatted in the same text style. Features 22 to 24 describe the number of pages which have footers present on them. Respectively, features 22, 23 and 24 are set to true if there are footers on over 25% of all pages, over 50% of all pages and over 75% of all pages. Features 22 to 24 are mutually exclusive.



- (25-28) Features 25 to 28 describe the geometric properties of the captions present in the document. If there are no captions present all these features are set to false. Otherwise feature 25 is set to true. Features 26 to 28 are mutually exclusive. Feature 26 is set to true if all the captions detected have the same text style and are in the same position relative to the image they are a caption of: above, below, left of or right of. Feature 27 is set to true if all the captions are formatted in the same text style. Feature 28 is set to true if all the captions are positioned in the same place relative to the image that of which they are the caption.
- (29-32) Features 29 to 32 indicate the number of different title block text styles present in the document. Respectively, features 29, 30, 31 and 32 are set to true if there are less than 4 title styles present, less than 9 styles, less than 20 styles and 20 or more unique title text styles. Features 29 to 32 are mutually exclusive.
- (33-36) Features 33 to 36 state the maximum number of columns found on any of the pages in the document. Respectively, features 33, 34, 35 and 36 are set to true if 1, 2, 3 and more than 3 columns is the maximum present in the document. All these features are mutually exclusive.
- (37) Feature 37 is set to true if the number of columns found on each page of the document differs from page to page. Newspaper documents generally have inconsistent column styles from page to page.
- (38-41) Features 38 to 41 describe the average number of images per page. They are mutually exclusive. Respectively, features 38, 39, 40 and 41 are set to true if the average number of images per page is 0, between 1 and 5, between 6 and 9, and greater than or equal to 10. Newspaper documents typically have a high frequency distribution of images per page.

- (42-45) Features 42 to 45 describe the geometric properties of the image blocks with respect to the columns present in the document. Feature 43 is set to true if there are images present which straddle more than one text column. Feature 44 is set to true if there are images present which are in line with a text column. Feature 45 is set to true if there are images present which are positioned to the side of a text column. Feature 42 is set to true if all the images are positioned in one of the styles described above. Newspapers generally have images which straddle text columns. Academic documents generally have images which are in line with the text columns.
- (46-49) Features 46 to 49 describe the graphic line operators present in the document. Respectively, features 46, 47, 48 and 49 are set to true if the average number of straight lines per page in the document is greater than 60, between 40 and 59, between 20 and 39, and less than 20. Typically form documents have a large average number of straight lines per page. All these features are mutually exclusive.

### 5.5.2 Knowledge source algorithms

The knowledge sources supplement their 'hard coded' knowledge with knowledge inferred from a single pass of all the geometric blocks the document contains. The knowledge sources find the geometric patterns which the document's blocks (which lie inside their domain of expertise) form and then infer specific block classification rules using a combination of the patterns they have detected and the hard coded knowledge they were given to start with. In this manner, the knowledge sources can efficiently classify geometric text blocks from a wide variety of document layouts, without the need for a vast data base of document layout knowledge.

Each knowledge source has an individual activation requirement and an individual set of algorithms to execute once it is activated. Within the blackboard architecture there is a paradigm of hierarchical abstraction. Knowledge sources at the highest level of abstraction work from the results and conclusions of the knowledge sources at lower levels of abstraction. All the knowledge sources work to a two pass system. On the first pass they examine all the blocks that they know are relevant to their domain of expertise. In between the first and the second pass they train themselves upon the data they have seen and devise rules, heuristics and statistical facts about the data. During the second pass they apply the rules and heuristics and tag the blocks that they had previously trained themselves on. After the second pass the knowledge sources examine the results of their tagging and infer a feature (or features) from the document which can then be used to classify the document with the help of a neural net.

The following sections document the algorithms that the individual knowledge sources use to train themselves, to tag blocks and to infer document features.

### **5.5.2.1 Block KS**

The Block Knowledge Source is responsible for creating the geometric blocks from the PDF document. The geometric blocks are the basic input to the system. The act of creating these blocks is document analysis.

The algorithms which describe the document analysis process are presented in section 5.3, “Final System Document Analysis of PDF”. The Block Knowledge Source executes these algorithms when it detects that the system is ready for more blocks.

The text blocks are created for every page within the document and then presented to the blackboard one by one. After a page of blocks has been presented to the blackboard (and thus the other knowledge sources) they are stored to disk using the storage system described in section 5.4.2.5.

Image blocks are easily found in PDF documents as they are already segmented and classified within the PDF document model. The Block KS will not allow any image block below a certain threshold in both the horizontal and vertical dimensions to be passed on to the analysis system as an image block. This is an attempt to stop smaller insignificant image blocks confusing the heuristics used by the Caption KS to identify caption blocks for bona fide images.

### 5.5.2.2 Text Frequency KS

The Text Frequency KS uses the information provided by the Block KS. It can be considered as existing on a higher hierarchical level than the Block KS. It studies every geometric block which is placed upon the blackboard and builds up a histogram of textual styles and the frequency of their occurrence within the document.

A 'style' is defined as a unique combination of font type and point size. The font is represented in the PDF document model (exported as an API by Adobe) as a font metric.

This knowledge source is also responsible for inferring the main text body style within the document. It does this using a recursive function upon the constructed histogram, after all the blocks in the document have been seen. The knowledge source expects at least one text style to be dominant but has the capacity to infer multiple main text styles depending upon the histogram and the pre-defined thresholding functions which it applies to the histogram.

#### Finding the main text styles

Figure 16 presents a flow chart which represents the main text style finding algorithm. The Main Text Knowledge Source executes this code in its training phase. The algorithm examines the histogram produced by analysing the text styles present in the document and the frequency of their occurrence. The frequency is calculated as a percentage of the document's total word content. The knowledge source initially sets the 'main text threshold' at 80%. If no text style is found which represents eighty percent of

the formatted text then the threshold is lowered by 5% and the algorithm is repeated. If a text style is found when the threshold has been lowered to less than or equal to 30% the knowledge source reasons that there may be another text style which could represent the main text of the document. It searches for the next highest text style in the histogram. Once it has found the next highest, it makes sure that this text style represents at least 20% of the text in the document before accepting it as a *bona fide* main text style.

### 5.5.2.3 Graphic KS

The Graphic KS simply looks at the number of graphical elements upon each page and calculates the average number of straight line graphic operators per page throughout the document. This simple document feature is the lowest abstraction that the graphical KS can infer. No details of how the graphical elements are used upon the page is inferred by either this knowledge source or the Block Knowledge Source when it segments the text blocks within the document. Graphical elements are used to create boxes in tables, define page boundaries and define article or text block boundaries within a document but none of this information is used by this system. This is not an oversight on behalf of the system implementor but a deliberate policy of not to be confused by a very complex area of document processing.

### 5.5.2.4 Structure KS

The Structure Knowledge Source plays a very limited role in the STASIS system. It has been incorporated into the system to provide the functionality that the Title Knowledge Source requires to find candidate titles based on a block's geometric 'influence' over other blocks. A Structure Knowledge Source would be fully utilised in a document understanding system. In such a system once the class of document is known the Structure Knowledge Source could apply class specific document structure finding algorithms.

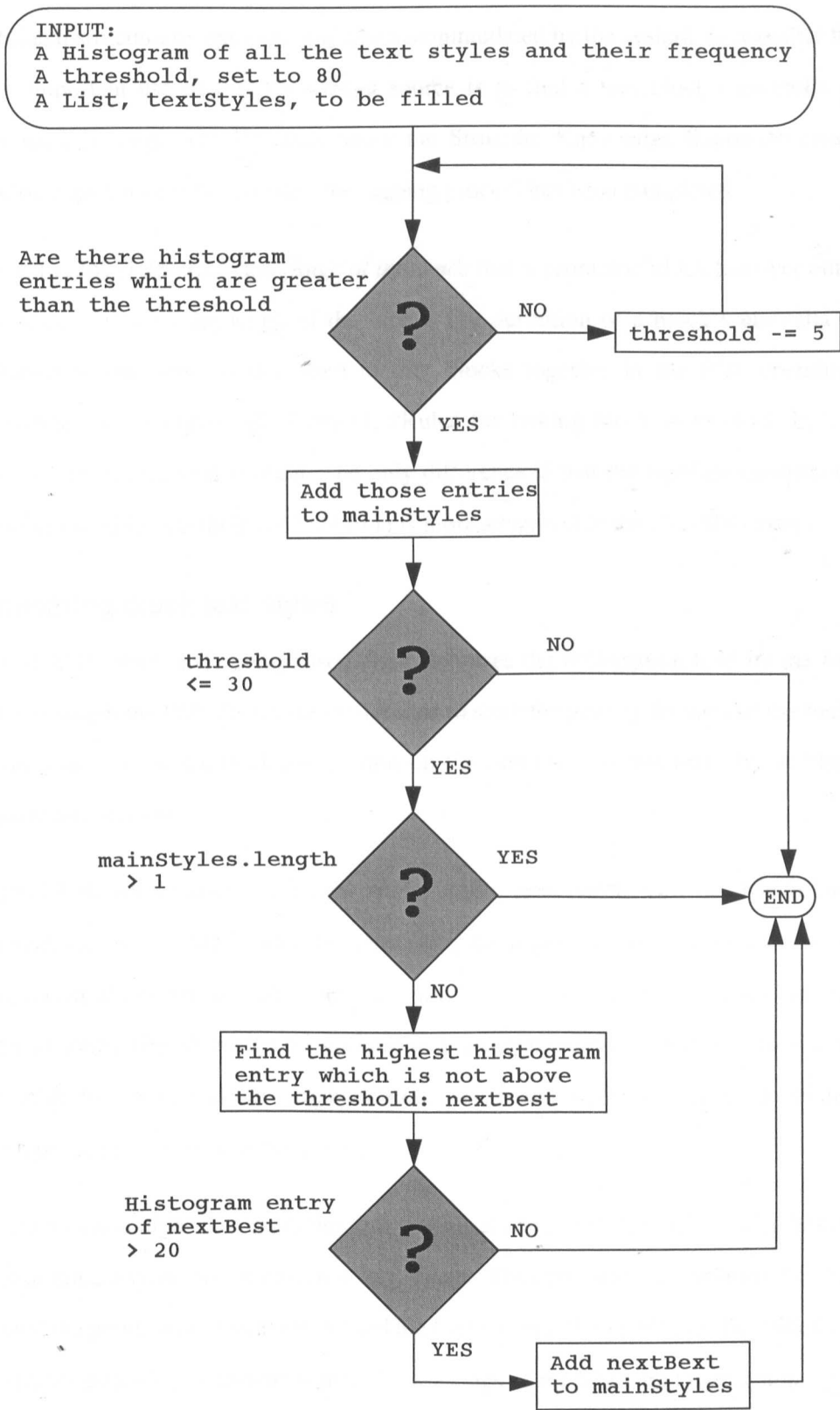


Figure 16: A flow chart of the main text style finding algorithm

The STASIS system was designed with a structure knowledge source so that any additional structure recognition could be accommodated by the system. In actuality the most important use of this knowledge source is to find a text block's umbrella of influence, although STASIS does allow the Structure Knowledge Source to create relationships between blocks after the tagging process has been completed.

Saitoh [Saito93] describes the *range of influence* that a geometric block has over other blocks as just being the width of the block. The definition of a block's umbrella of influence is the same as that used to link blocks together in the PDF document understanding prototype, see Table 11, "Rules for linking block A to block B," on page 101 for a precise definition. The only difference is that the typeface comparison algorithm used in rule three (of Table 11) is more advanced in the STASIS system.

### Comparing block text styles

The STASIS block style comparison algorithm uses the information held by the font metrics, which the PDF document model uses to store the generic features of the fonts. A font metric does not hold information on the point size of the text; this is block specific information.

Figure 17 shows a variety of letters, all set in the *same* point size, but which have different metrics. Figure 17 identifies the component parts of the metrics used in the comparison algorithm and illustrates the variance of some of these metrics between different fonts. The advantage of working with the font metrics, rather than font name and point size, is that a more sophisticated comparison algorithm can be developed which produces a more accurate result.

The comparison algorithm examines two text blocks and finds the block which has the more dominant geometric appearance on the page. The comparison algorithm takes into account the point sizes of both blocks and the font metrics of both blocks. The following paragraphs describe, in abstract terms, the workings of the block text style comparison algorithm.

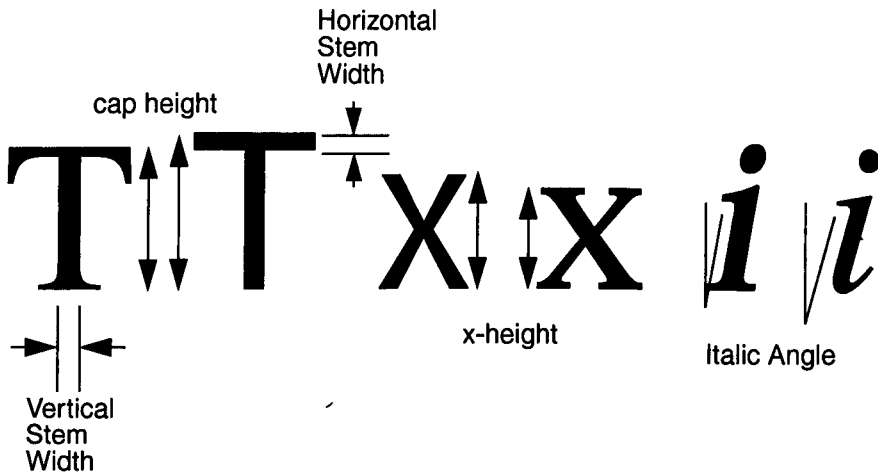


Figure 17: Parts of characters

If both blocks have identical metrics (in other words they have exactly the same font) the point sizes of the text blocks are used to differentiate the blocks. If the metrics are not identical, then two confidence values are set up. One value is assigned to each text block. Various individual comparison tests are carried out on the metrics and the appropriate confidence is incremented by a pre-calculated amount depending on the results. By incrementing the confidence value of a block the algorithm literally increases the confidence that that block is the more geometrically dominant of the two.

The individual comparison tests examine the font attributes of the two blocks (some of which are illustrated in Figure 17), for example cap height. If block A has a greater cap height than block B, then the confidence value assigned to block A is incremented by 0.2 whilst the confidence value of block B remains the same (and vice versa). The magnitude of the increment is proportional to the contribution which that attribute makes to the geometric appearance of the font. The same process is repeated for the following attributes of the font metrics (the confidence increment magnitude is shown in brackets after the attribute): vertical stem width (0.5); horizontal stem width (0.5); x-height (0.2); italic angle (0.2) and point size (0.6) although point size is an attribute of the text block and not the font metric.



The confidence value which is greatest at the end of all these tests represents the geometrically dominant text block. A block must be geometrically superior than the block(s) over which it has a geometric influence.

Locating blocks in the umbrella of influence

Figure 18 shows various examples of blocks inside and outside of an umbrella of influence. A block is said to have an umbrella of influence if it has an influence over one or more blocks. An analogy that may be useful is that the block can be considered as a rain cloud which is raining onto the blocks beneath it. If a block can feel rain on its head then it is under the influence of the block which is raining. The rain cannot pass through the blocks. Refer to Table 11 on page 101 for a precise definition of the rules used to link blocks together in the STASIS system.

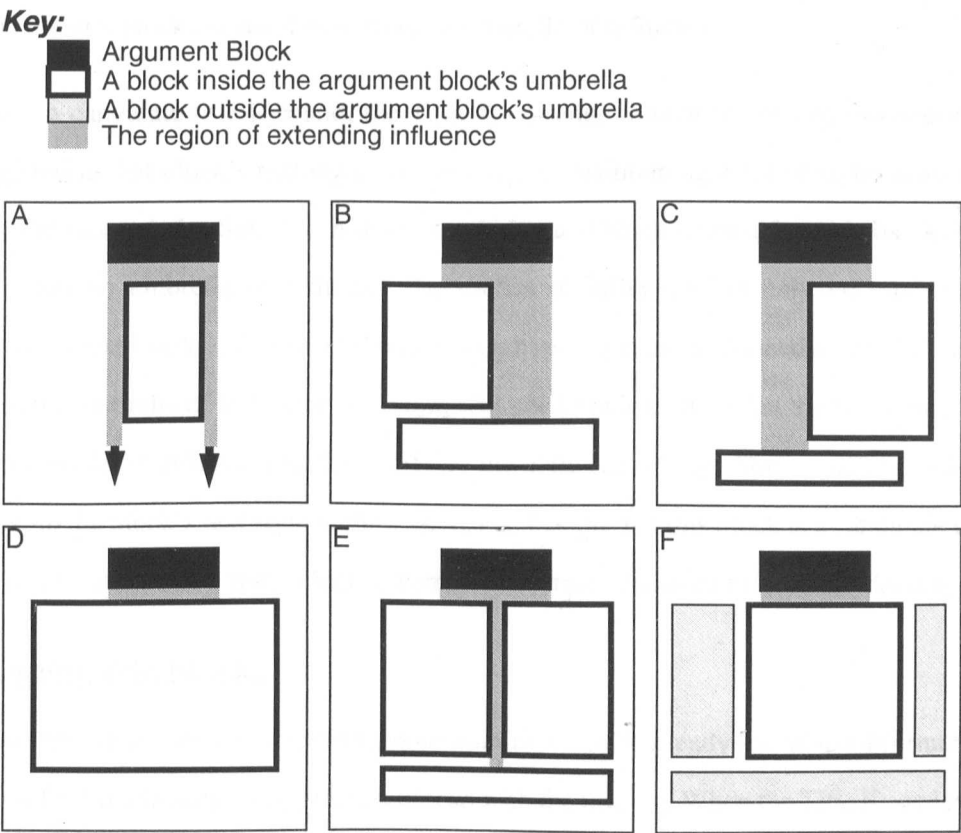


Figure 18: Various umbrellas of influence

### 5.5.2.5 Title KS

The job of the Title KS is to infer which blocks are potential titles within the document. The Title KS examines every text block which is placed upon the blackboard and extracts as much information as it can from the blocks it has seen. The Title Knowledge Source initially establishes whether or not a given block has an umbrella of influence over other blocks on the same page. It does so using functionality obtained by inheriting from the Structure Knowledge Source class. The Structure Knowledge Source can determine if a given block has an umbrella of influence by examining that block's position on the page and the position of other blocks beneath it.

If the Title Knowledge Source finds a text block with an umbrella of influence it tags that block as a 'candidate title'. This is so that when the Title Knowledge Source has trained itself and is ready to tag blocks with the 'title' tag it need not re-test the document's blocks to see if they have an umbrella of influence.

Once a candidate title is found, the Title Knowledge Source records its discovery by registering that block's text style. The purpose of maintaining a list of styles is so that the Title Knowledge Source can still reason that a text block is possibly a title block even if it has no umbrella of influence. Umbrellas of influence can only be established between text blocks. If a *bona fide* title block happens to be positioned above an image then the umbrella of influence algorithm will not be able to state that the title block has an umbrella of influence. In this situation the Title Knowledge Source will be able to look up the block's text style in the style list and assert that this block is a candidate title as it is formatted in a style which is common to other candidate titles in the document.

#### Tagging title blocks

The Title Knowledge Source requires no training stage. It already has all the information it can find at this stage to tag the text blocks with the title tag. When the Title Knowledge Source examines a block during the labelling stage it checks to see if the block is tagged with the 'candidate title' tag. If the block is appropriately tagged the Title Knowledge Source asserts that the block is a *bona fide* title with a confidence of 0.4.

## Final System Development and Engineering

---

The magnitude of the confidence of this assertion was set after inspecting how the title knowledge source and other knowledge sources behaved during tests on trial documents. 0.4 was chosen because it accurately represented the confidence that the Title Knowledge Source had in its assertion (at this point in the system execution cycle). The process of setting and adjusting the confidence values of the assertions made by all the knowledge sources fall into the category of system engineering. Considerable care was taken *not* to engineer the system to assign too *great* a confidence value to an assertion after examining weak evidence for that assertion and similarly, care was taken *not* to let the system assign too *small* a confidence value to an assertion after examining strong evidence for that assertion.

If the text block being examined is formatted in a text style which the Title Knowledge Source recognises as one of the more prominent text styles in the document, the confidence of the assertion is increased by 0.1. The Title Knowledge Source 'recognises' the text style by looking up the font metrics associated with the text block style and comparing them to the list of candidate title text styles which it collected as it was being presented with the training data. Prior to the labelling stage, the Title Knowledge Source ranks the list of styles in a decreasing order of geometric prominence. If the text style is placed in the top third of the list, the confidence of the assertion is increased by 0.1 again.

This algorithm is designed to decrease the chance of a document header being mis-tagged as a title. In heuristic terms, the knowledge source 'knows' that headers are unlikely to be formatted in a text style which is one of the document's most prominent geometric styles.

The magnitude of these confidence values is assigned after a process of trial and error. During the trials, test documents are fed to the system and the results of the tagging process are recorded. The accuracy and reliability of the systems classifications are assessed and the confidences adjusted so that they are a true reflection of the confidence of the systems assertions.

### **Resolving conflicts with other knowledge sources**

The Title Knowledge Source is the only knowledge source which requires knowledge of a geometric text block's context on a page before it can assert a statement with any confidence about that block's eligibility to be a title. The Title Knowledge Source needs to know the features of the blocks which can be found beneath the block which the Title KS is considering to tag; paramount among the required features of the surrounding blocks are their tags.

During the labelling stage of advanced document analysis, if a conflict arises between the Title Knowledge Source and any other knowledge source(s), the Title Knowledge Source does not have all the information available to make a confident assertion instantly, as the blocks beneath the block being considered by the Title Knowledge Source have not yet been tagged. In this situation, the controller records a 'blue print' of the conflict. The controller records the knowledge sources which are conflicting and the block which they are conflicting over. The controller pushes this information onto a stack data structure. Once this is done the controller allows the tagging process to proceed with the next block.

Once all the blocks (which can be tagged without involving a Title Knowledge Source conflict) on a single page have been tagged the controller processes the recorded conflicts which involve titles. Title Knowledge Source can then use more detailed heuristic routines which use contextual information provided by the blocks which surround the candidate title block being tagged. Appropriately, the controller then asks the Title Knowledge Source to re-submit its assertion with a newly calculated confidence.

The confidence of the assertion made by the Title Knowledge Source depends on the tags of the blocks lying around it on the page. The Title Knowledge Source is most confident when the candidate title block lies over a main text block. The Title KS is less confident when the candidate title block lies over another title and the least confident when the candidate title block lies over an unknown block.

The controller records the conflicts on a stack and then processes the conflicts by popping the stack. The blocks on the blackboard are always ordered so that the first blocks encountered are the blocks at the top of the page. By maintaining a stack of the conflicts and pushing new conflicts *onto* the stack, the controller can work up the page by processing and popping the conflicts *from* the stack. By working up the page the controller can tag blocks which may be under other candidate title block's umbrellas of influence. This ensures that the controller never has to ask knowledge sources to make assertions about untagged blocks based on the status of other blocks which are waiting to be tagged.

### 5.5.2.6 Super-title KS

The role of the Super-title KS is unique among the knowledge sources in that it is hard coded with a number of specific heuristics to find the title of the document currently being processed. The super-title of a document can be easily deduced by analysing the text styles within the document and finding the largest and most geometrically prominent. If this style occurs only once in the document and this occurrence is on the first page then that block is tagged as a super-title. The confidence of this assertion can be made relatively high as document layout designers normally always make the document title the largest piece of text on the first page of the document. The Super-title KS has a set of subsidiary heuristics which allow the super-title to link to other text blocks on the first page which lie in close proximity to it. In this way the full title of the document can be protected from the remaining advanced analysis algorithms.

### 5.5.2.7 Image KS

The Image KS analyses every image block which it sees upon the blackboard. It then calculates the average number of images per page throughout the document.

### 5.5.2.8 Caption KS

The Caption KS communicates closely with the Image KS to find captions for the images in the document. The Caption KS obtains a list of all the images on a page prior to analysing the text blocks upon that page. Then it stores a copy of every text block which is within a certain distance of any image block. That distance is calculated for each image as being 30% of either the height or width of the image.

The 30% threshold was implemented to speed up the calculations made by the Caption KS by reducing the number of candidate captions it analysed. A value of 30% was chosen after several tests were made on sample documents. The tests ensured that the threshold was great enough to contain all the credible candidate captions (including the bona fide caption) and yet small enough to exclude the improbable caption blocks.

After all the blocks in the document have been seen once, the Caption KS attempts to find patterns amongst the potential caption blocks it has collected. For every image on the page all the blocks which lie within the 30% zone around that image are placed into one of four categories. One category for each direction away from the image: north (up); south (down); east (right) and west (left). The physically closest block to the image within a specific direction category is retained whilst all others are discarded. Potential caption blocks are also discarded if they have a greater area than the image to which they are associated with. The result is potentially four (or less) blocks which pertain to being that image's caption. This algorithm is executed for every image in the document.

The Caption KS then looks for recurring patterns amongst the blocks that are associated with the image. Two types of patterns are looked for: position and style. If every image in the document (which has at least one potential caption) has a potential caption in a particular dimension (for example, below the image) then the Caption KS can assume that this is a *bona fide* choice of style from the author (or designer) of the document. If every image has a potential caption which has a particular text style, then the Caption KS can assume that this is another *bona fide* choice of style from the author of the document.

The Caption KS will calculate the confidence in its self inferred heuristic from the percentage of image blocks which conform to the heuristic, for example, it will have 100% confidence if all images conform and 50% confidence if only half do. In situations where the only potential captions for an image conform to neither a position based heuristic or a style based heuristic, the potential captions are ranked according to their physical proximity to the image. The creation of these heuristics takes place in the training phase of the Caption KS life cycle. When the Caption KS wants to tag a particular block with a caption tag, the confidence with which it hints to the controller that it wishes to tag the block is directly proportional to that blocks conformity to the self generated heuristics.

After tagging, the Caption KS can state concrete facts about the captions present in the document based on style and position pattern recurrences. The Caption KS can easily detect and analyse the consistency of the recurring caption styles.

### 5.5.2.9 Footer KS

During training, the Footer KS examines every block which it sees on the blackboard and retains a copy of the blocks which lie in the bottom  $\frac{1}{8}$ th of the page. It uses this information to generate heuristics based on inferred patterns about the geometric properties of the blocks which it has seen. The training and tagging routines are abstracted to the Peripheral KS class from which the Footer KS inherits.

### 5.5.2.10 Header KS

The Header KS operates in a similar manner to the Footer KS, except that it retains a copy of all the blocks which lie in the top  $\frac{1}{8}$ th of the page. The Header KS also inherits functionality from the Peripheral KS class.

### 5.5.2.11 Peripheral KS

The Peripheral KS is the base class for both the Footer and Header Knowledge Sources. Many of the pattern finding algorithms which are used to locate potential headers and footers are abstracted into this class. The Header and Footer KS classes simply collect the blocks which could form part of a header or footer. The Peripheral KS looks for blocks which recur in approximately the same position page after page and have the same geometric style. The confidence in its findings are based on the length of the document and the regularity of the pattern recurrences. The Peripheral Knowledge Source knows about relative peripheral positioning upon a page (most headers and footers either occur left, right or centre justified).

No knowledge about the content of the peripheral blocks has been incorporated into the system because of the desire to leave the system language independent. Furthermore, this type of association (i.e. realising that the text content of a header relates to the text content of the current section heading or news article) is typical of document understanding algorithms and so it does not need to be realised at this point in the document processing cycle.

The Peripheral KS can make a guess as to whether the document is single sided or double sided based on peripheral geometric block analysis. By assuming that the document is single sided the Peripheral KS risks creating a weaker heuristic for peripheral tagging than if it considers the possibility that the document is double sided.

Once the Peripheral KS has finished creating heuristics (and the confidences in those heuristics) for a single sided document, it can explore the possibility that the document is double sided in the following manner. By taking the blocks which it knows about from every other page and reflecting them in the vertical axis which runs down the centre of the page, the Peripheral KS has in fact taken a double sided document and transferred the coordinates of the peripheral blocks to represent a single sided document. If the heuristics (created from analysing this new transformation upon the peripheral blocks)



are stronger than the heuristics created from analysing the document when it was assumed that the document was single sided, then it is reasonable to suggest that the document is double sided.

### The Peripheral KS tagging routine

Peripheral KS calculates how much it believes a block to be a *bona fide* peripheral block based on the analysis of the length of the document and the number of lines in the block being tagged. This code is invoked by either the Header or Footer Knowledge Sources once it has seen a block on the blackboard (during the labelling stage) which is positioned in the appropriate part of the page: either the top or bottom  $\frac{1}{8}$ th of the page respectively.

The assertion confidence is given an initial value of 0.1 because the block is placed in the correct part of the page to be a peripheral. 0.55 is added to the confidence if the position and style of the block matches that of a pattern which has been inferred by the Peripheral KS as possibly being a header or footer pattern.

The assertion confidence is then either added to or reduced based on the length of the document. This qualifies the reliability of the pattern finding algorithms. If the document is short then the pattern finding algorithms will not have had much input data to work with; if the document is long then the pattern finding algorithms can be trusted more.

Finally the peripheral knowledge source looks at the number of lines in the argument block. If there is only one line in the block the assertion confidence is increased by 0.1; if there are two lines in the block the assertion confidence remains the same. From then onwards, for every extra text line detected 0.1 is knocked off the assertion confidence. These statements are based on the assumption that the more lines there are in the block the less likely it is to be a header or a footer block.

### 5.5.2.12 Column KS

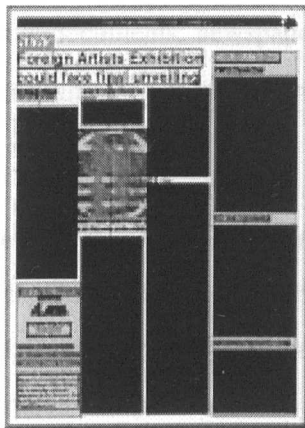
The Column KS finds the location of text columns on a page. The high level features that can be inferred from such knowledge include the number of columns per page, the consistency of the column format throughout the document and the consistency of the position of the images in the document with regard to the columns. The Column Knowledge Source can tell if the images in the document straddle more than one column or if they are always in line with the bounding boxes of the columns or if they are to the side of the bounding boxes of the columns. It can also tell if any of the formatting styles mentioned above exist together in a document.

The Column KS uses the information found by the Block KS and the Text Frequency KS to extract all the body text blocks from a document after the process of tagging. Left with only the text blocks it performs a vertical projection of their bounding boxes to create a histogram. Then by analysing the resulting histogram for maximas and minimas the KS can accurately estimate the number of columns and their bounding boxes.

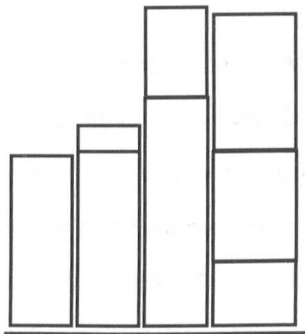
Figure 19 (a) shows an actual page of a newsletter which has been decomposed by the STASIS system. The boxes which have been tagged as main text blocks have been covered with a black rectangle by hand. Figure 19 (b) shows the bounding boxes of the main text blocks stacked up on one another to form a histogram. The dimensions of the columns can be deduced from this histogram.

### 5.5.2.13 Document Class KS

The Document Class KS takes all the features that the other knowledge sources have inferred from the document's geometric properties and classifies the document into one of four different classes. It accomplishes this using a pre-trained neural net. This is explained further in the next section.



(a)



(b)

Figure 19: A decomposed page and its column histogram

# 5.6 Development of a document classifier

The blackboard system can extract information from a document which describes the geometric properties and to a limited extent the logical properties of that document. This information is extracted as document features. These features allow the STASIS system to classify documents into logical classes. When used without the other, neither geometric features nor basic logical features provide enough information to accurately classify a document.

The system has been engineered to classify a geometric document into one of four general logical classes: academic documents; newspaper documents; form documents and brochure documents. The attributes of these document classes are described in more detail in the next section.

The working hypothesis of the system is that documents from unique logical classes, which share the same base logical class, will exhibit similar logical traits on a fundamental level, for example, newsletters, newspapers and magazines are documents which all have basic logical features which are common to a base class which describes documents as consisting of articles. The STASIS system can identify some of their basic logical features as well as their common geometric features.

Whilst designing the system, it was hoped that STASIS would be able to classify documents into a fifth logical class: business letters. Business letters have no distinct basic geometric features or distinct basic logical properties. STASIS could have been re-programmed to detect high level logical business letter features such as addresses and signature, in a similar manner to Lam's model-driven classification approach [Lam94a]. The re-programming would involve the creation of model-driven routines which would be specifically designed to detect business letters and which could not be used on other types of document. This solution was rejected in order to keep STASIS' classification strategy purely data-driven.

### 5.6.1 Specification of the target logical document classes

A description of the target logical document classes is presented in this section together with the detectable geometric and logical attributes associated with each class. All the attributes listed in the next four sections are **not** always present in **every** instance of each document class. Some documents may contain only a sub-set of these features.

### 5.6.1.1 Logical newspapers

The logical newspaper class is the base class for broadsheet newspapers, magazines and newsletter documents. Each of these documents can be described by a basic logical document class which consists mainly of logical articles. Logical articles consist of article title blocks, paragraph blocks and logical sub-stories. Sub-stories consist of sub-story title blocks and more paragraph blocks and potentially more sub-stories. This definition is closely related to the definition presented by Niyogi in Table 7, on page 84. The general characteristics of this class of documents include:

- a high proportion of images per page;
- multi-column page layouts which, with the exception of newsletter documents, vary from page to page;
- a double sided layout;
- a large number of text styles containing a large number of title styles;
- possibly more than one style of main text.

### 5.6.1.2 Logical academic documents

The logical academic document is the base class for all documents which are partitioned into chapters, sections, subsections and so on. Books (fiction and non-fiction), journals, technical documents and conference proceedings are specific examples of this type of document. General characteristics of this class include:

- few text and title styles of which only one text style is the main text style;
- a consistent page layout style which is typically single columned (although many papers and journals are double columned);
- headers and footers which are formatted in a consistent style;
- consistent caption styles.

### 5.6.1.3 Logical brochures

A logical brochure is a document which is devoted to a single topic matter. It is the container of a single logical article. Although there is a certain amount of similarity between the logical classes of the newspaper and brochure classes, the geometric and base logical features found in both classes are diverse enough to confuse the document class recognition routines if they were grouped into one target class. Brochure documents typically exhibit the following features:

- a complex page layout similar in style to a newspaper;
- very short in length. This causes the absolute number of text and title styles to be closer to the characteristics of an academic document rather than a newspaper;
- a large average number of images per page.

### 5.6.1.4 Logical forms

Logical forms represent documents in which the majority of the document is formatted as a table. The table can either be empty or full of data. Typical examples of documents in this class include tax forms, questionnaires and invoices. All of these documents exhibit the following characteristics:

- fairly short documents which are rarely over ten pages in length;
- few text styles and few title styles;
- hardly any images and an irregular column style caused by the absence of any substantial main text blocks;
- an extremely high number of straight lines per page. This is arguably the most important feature STASIS extracts from form documents.

### 5.6.2 Classification techniques

So far this chapter has described the techniques used to extract meaningful features from a PDF document and has described the desired target categories of the classification process. The following section summarises the various classification techniques that were considered and justifies the eventual choice of a neural net classification system.

#### 5.6.2.1 Basic classifier requirements

The STASIS system is designed to classify any document into one of four logical categories based entirely on data generated from the analysis of a document's geometric properties. In practice this is a problem which does not have a sound solution. The STASIS system is reliant upon the document author's judgment (and the judgement of the author of this thesis) of document formatting and document style. Furthermore, certain documents do not have geometric features which are *unique* to their class of logical document. There is, unquestionably, a large amount of 'uncertain' data present in the STASIS system. The classification technique must be able to handle uncertain data.

The magnitude of the document classification problem demands that the classification algorithms must learn from the data presented to it and be flexible enough to recognise a wide variety of document features of which any one feature, or any combination of features, can provide vital classification information for one class of documents, but be useless or ambiguous for another class of documents.

#### 5.6.2.2 Production rule expert systems

Expert systems are programs which specialise in solving problems within a certain domain of expertise. The name 'expert system' reflects the fact that these systems are usually based on knowledge obtained from people who are experts in a specified discipline [Gonza92, Jacks92]. Many expert systems employ a rule based deduction system, which stores knowledge in the form of rules which follow the syntax: *if X then Y*.

Typically, the rules are fired in either a forward chaining strategy or a backward chaining strategy. Forward chaining is the process of moving from the *if* patterns of the rules to the *then* patterns of the rules, using the *if* patterns to identify situations for the deduction of a new assertion. Alternatively, the backward strategy initially forms a hypothesis and uses the knowledge encoded in the rules to work backwards to find assertions which support the hypothesis [Winst92].

Production rules are usually created for systems which seek to emulate human decision making processes. Human document class recognition probably relies upon a combination of cognitive analysis of the page layout plus semantic knowledge of the document content. Humans are good at identifying documents which have an easily identifiable page layout. Humans can identify newspaper documents as newspapers simply because of the style of the document regardless of the language the newspaper is written in. Documents which do not have such a unique and easily identifiable appearance require a certain degree of semantic processing in order for humans to successfully classify them. The STASIS system performs no semantic processing of the logical content of a document. This was a deliberate decision made in order to keep the system relatively language independent. Consequently, the STASIS system is deprived of a large aspect of human cognitive decision making.

Niyogi's document understanding system is based on a production rule system [Niyog94, Niyog95, Niyog96]. Niyogi's system does not address the problem of global document classification and it restricts itself to the classification of newspaper documents. A production rule system for global document knowledge would be an enormous system with many rules at a variety of different levels of abstraction. The task of maintaining and engineering such a system would be equally large. 'Engineering' a production rule is the term used to describe the tuning of the rule's uncertainty factors. Uncertainty factors represent the same thing as confidence factors: the magnitude of confidence an expert system has in an assertion. Uncertainty factors are a good way of modelling uncertain data in systems in which a diagnosis is being performed. By using uncertainty factors, the production rule system can apply rules which will lead the system down a certain avenue of investigation.



There are several techniques for modelling uncertainty in production rule systems. Each has its own advantages and disadvantages and each is suitable for one type of problem solving and not for another. Leaving aside symbolic uncertainty reasoning, which is inappropriate given the size of the problem and the number of features involved (see section 5.4.1.5, “Example blackboard systems” for an example of symbolic uncertainty reasoning), the most popular numerical methods of modelling uncertainty in production rules are Bayesian probability [Stutz94] and fuzzy logic [Zadeh83].

From a mathematical perspective, fuzzy sets and probability exist as parts of a greater generalised information theory which includes many formalisms for representing uncertainty (including random sets, Demster-Shafer evidence theory [Shafe76], probability intervals, possibility theory, general fuzzy measures and interval analysis, to name a few) [Jacks92].

In semantic terms, the distinction between fuzzy logic and probability theory has to do with the difference between the notions of probability and a degree of membership. Probability statements are about the likelihoods of outcomes. Fuzzy logic cannot say unequivocally whether an event occurred or not and instead tries to model the *extent* to which an event occurred [Klir92].

The largest problem in using a rule based system for classifying documents is that it is not obvious how humans cognitively classify documents. There is no model which can be used to help create the knowledge rules. Famous expert systems such as MYCIN and Prospector have solved problems in domains of expertise in which the translation of human knowledge to production rules has been straight forward [Jacks92].

The task of creating the knowledge rules, ‘tuning’ them with an uncertain data handling strategy and constantly modifying them with knowledge built up from processing examples is difficult enough when the classification problem space is easily defined and explained by human experts. As the classification problem the STASIS system is attempting to emulate is cognitive in nature, it was felt that the level of ‘knowledge engineering’ required to competently complete this problem was too high. Consequently, a production rule classification system was not chosen.

Instead STASIS uses a neural net. Neural nets are based on biological neural connections and are proven to be able to 'see through' uncertain data and extract underlying patterns. However, STASIS is liable to the disadvantages of a neural network. A neural network builds up its classification abilities through training cycles using examples which it assumes are correct. The document classification problem is difficult to rationalise. If the neural net is trained with features extracted from documents which one believes exhibit the geometric and logical properties of a specific class of documents, but which in actuality do not (or worse they exhibit the properties of a different class), then the neural net will have been badly trained and documents will be incorrectly classified.

### 5.6.2.3 Neural Net Classifiers

Artificial neural networks were inspired by the elementary functions of the biological neuron. They can modify their behaviour in response to their environment. Shown a set of inputs and desired outputs they can self adjust to produce consistent responses [Wasse89]. Neural networks display a surprising number of the brain's characteristics, for instance, they learn from example, they generalise from previous examples to new ones and they abstract essential characteristics from inputs containing irrelevant data.

Once trained, a network's response can be (to a certain degree) insensitive to minor variations in its input. Networks have the ability to see through noise and distortion to the pattern that lies within the input. This ability makes the neural network applicable to real world pattern recognition problems.

Multi-layer networks have computational abilities beyond single layer networks [Wasse89]. Multi-layer networks are formed by cascading a group of single layers. The output of one layer provides the input to the subsequent layer. The set of inputs  $\mathbf{X}$  has each of its elements connected to each artificial neuron in the first layer with a separate weight. The weights should be considered as elements of a matrix  $\mathbf{W}$ . The dimensions of the matrix are  $m$  rows by  $n$  columns, where  $m$  is the number of inputs and  $n$  the

number of neurons. Figure 20 shows a two layer neural network. The circles on the left hand side of the diagram represent inputs. The squares are working neurons. Figure 20 is a replication of a diagram used by Wasserman [Wasse89].

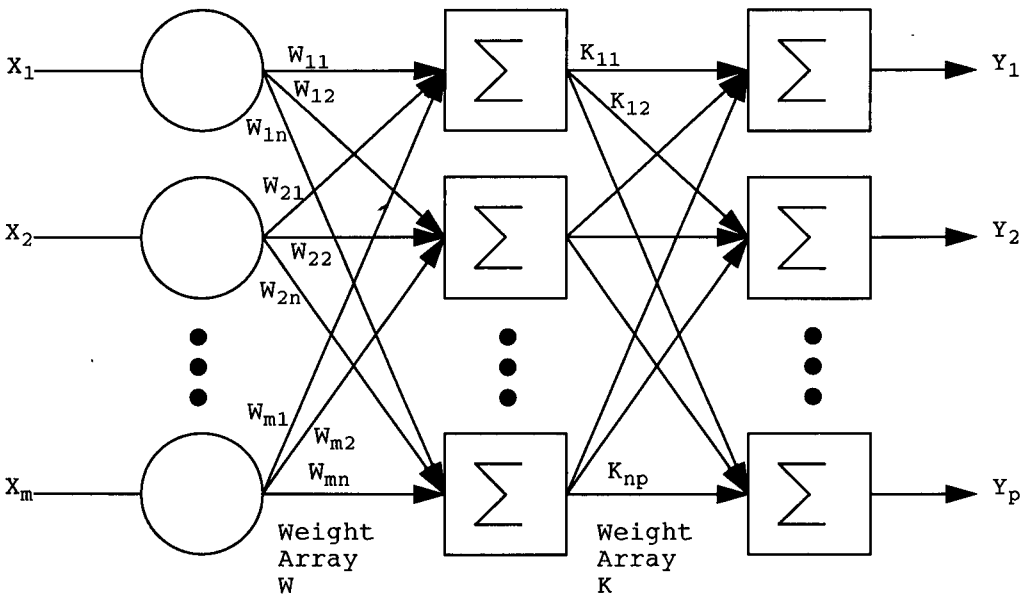
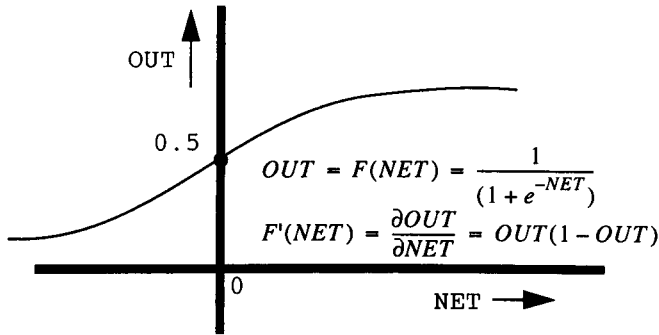


Figure 20: A two layer neural network

Each neuron sums the products of all the input units and their weights to produce an output value that is known as NET. The net signal is further processed by an activation function to produce an output signal, known as OUT. Single layer networks have linear activation functions. Double layer neural nets require non-linear activation functions in order to prevent them being restricted to the computational capabilities of a single layer network.

Before the invention of the backpropagation algorithm there was no theoretically sound way of training multi-layer neural networks. The backpropagation algorithm uses the Sigmoid activation function. The Sigmoid function compresses the range of NET so that OUT lies between one and zero. For small signals (NET near zero) the slope of the input/output curve is steep which produces a high gain. For large values the gain decreases. Thus large signals can be accommodated without saturation and small signals are allowed to pass through without excessive attenuation. Figure 21 shows the Sigmoidal activation function.



*Figure 21: The Sigmoidal Activation Function*

The backward propagation algorithm is well documented by Wasserman [Wasse89] and Winston [Winst92]. A brief description of the principles of the backward propagation algorithm follows.

The back propagation training method computes the changes to the weights in the final layer first, reuses much of the same computation to compute changes to the weights in the penultimate layer and ultimately goes back to the initial layer. Back propagation makes a large change to a particular weight,  $W$ , if the change leads to a large reduction in the errors observed at the output node. For each input combination the output's desired value is considered,  $d$ , together with its actual value,  $O$ , and the influence of a particular weight,  $W$ , on the error,  $d - O$ . A big change to  $W$  makes sense if that change can reduce a large output error and if the size of that reduction is substantial. If a change to  $W$  does not reduce any large output error substantially, little is done to change  $W$ .

Like the human brains they mimic, neural nets retain a degree of unpredictability. Unless every possible input is tried, there is no way of being certain of the precise output.

The neural net used in the STASIS system was developed by Professor D. G. Elliman at the University of Nottingham. It is a multi-layer network which uses the backward propagation technique of pattern recognition together with the Sigmoid non-linear activation function.

**5.6.2.4 Development of the STASIS neural net classifier**

The neural net employed by STASIS has four output nodes (corresponding to the four target document classes), 49 input nodes (one node for each component of the input feature vector, described in Table 15 on page 145) and 20 nodes in the hidden layer. The magnitude of the hidden layer was arrived at by selecting a value approximately equal to the mean of the input and output layer magnitudes.

The neural net was trained manually using documents whose features were characteristic of their logical class. Table 16 lists the number of different documents from each target class which have been used to train the net.

Forms have been trained with relatively few documents. Forms have dominant geometric characteristics, thus the neural net requires few example form documents in order to establish the recurring characteristics of form documents. More importantly, it became clear during the engineering of both the neural net classifier and the advanced analysis feature extraction system that form documents were not being processed efficiently. There are a combination of factors which are present in form documents which produce an ambiguous set of document features. For this reason it was decided to suspend further training with form documents.

| Document Class | Number of Examples Used |
|----------------|-------------------------|
| Form           | 6                       |
| Newspaper      | 40                      |
| Academic       | 54                      |
| Brochure       | 56                      |
| Total          | 156                     |

**Table 16: The number of documents used to train the STASIS classification net**

Certain types of forms (typically those which consist of only single table with many graphic lines) are classified well by STASIS. However, the vast majority of form documents have explanation notes, or guidelines, in the document. The more regular

text the form document contains, the more STASIS believes it is processing a brochure or an academic document. Furthermore, it could be argued that the document features which STASIS has extracted from form documents contain ambiguous and contradictory patterns. A possible solution to this problem is the creation of good graphic analysis routines which can detect specific graphical features present in forms: boxes, corners and so on. This would help locate specific features and thus help the classification process. Forms are not well suited to the feature extraction and document classification strategy utilised by STASIS and should, instead, be detected by a model-driven classification strategy such as that presented by Lam [Lam94a, Lam95].

### The train and test cycle

Initially the neural net was trained with a couple of documents from each target class. These documents were chosen because they displayed the characteristics of their class. This established a balanced foundation for the classifier. This foundation was built up by downloading and training the neural net with PDF documents from the internet. The use of working documents in the training process gives some credibility to the classification results.

Each PDF document was processed by STASIS and then classified by the neural net. The full results of the neural net classification (including a list of all the net's output nodes final weights) were written to the document report (see section 5.2.1, "The STASIS interface" for more details on the document report). Generally, the results of the classification fell into one of three categories. Depending on the category, a different training strategy was adopted.

In the following category descriptions, the term 'unambiguous' refers to the magnitude of the difference between the final weights of the output nodes. A small difference between two nodes can be interpreted as an ambiguous decision even though one of those nodes may be numerically larger

- **Category 1.** The document classification was correct and unambiguous. In this situation the neural net was not re-trained.

- **Category 2.** The document classification was correct but ambiguous. If the report listed good characteristics for the class of document being processed, the net was trained with the features of this particular document appended to the list of features stored by the net for training purposes. If the report listed poor or uncharacteristic features, no action was taken.
- **Category 3.** If the classification was incorrect. In this case the document report was examined. If the report contained features which were *not* characteristic of the logical class of the document being processed, no action was taken. Training the neural net with misleading data would reduce the efficiency of the net. The confidences in its classifications would drop as the error values calculated in the classification process would increase. If the report contained features which *did* display characteristics of the desired logical target class it was assumed that the net had mis-classified because it had not seen an example document of this particular kind before; the net was duly trained with the new document features.

## 5.7 System output

Having classified the document, the system constructed its own internal representation of the document using a tree structure which contains nodes which are logical (for example title nodes) and nodes which are geometric (for example, header nodes). The structure contains geometric pages, logical links, geometric blocks and a logical document classification, all of which are held in a tree structure which when traversed gives an approximate reading order of the document. The reading order is not a true representation of a logical document's reading order, particularly if that logical document is a newspaper. Newspapers have complex geometric structures in which logical articles normally span one or more pages. The reading order the STASIS system provides is a reading order that one would find if one were to read a single page at a time. Even this is not complete; for example, academic journals typically have footnotes.

Logically, one would read the footnote when the reference is made to it in the main text body. Footnotes are formatted at the bottom of pages, therefore they will be ‘read’ last by the STASIS system.

Figure 22 illustrates the data structure which STASIS builds after completing its processing. The document class attribute is associated with the document root and is identified as an attribute by its rectangle box. The arrows should be described as ‘possibly link to’. Images ‘possibly link’ to captions but they do not have to. Captions on the other hand can only exist if they are linked to by images. The STASIS data structure does allow a title to link to other titles and so create a chain of titles. A title block in the chain will always have a lesser geometric text style than its parent title block and equally, it will have a greater geometric text style than its children. In order to create a link to a child, the parent title block must establish an umbrella of influence over the potential child block. An umbrella of influence demands that the geometric text style of the child block is geometrically inferior than their parent’s text style.

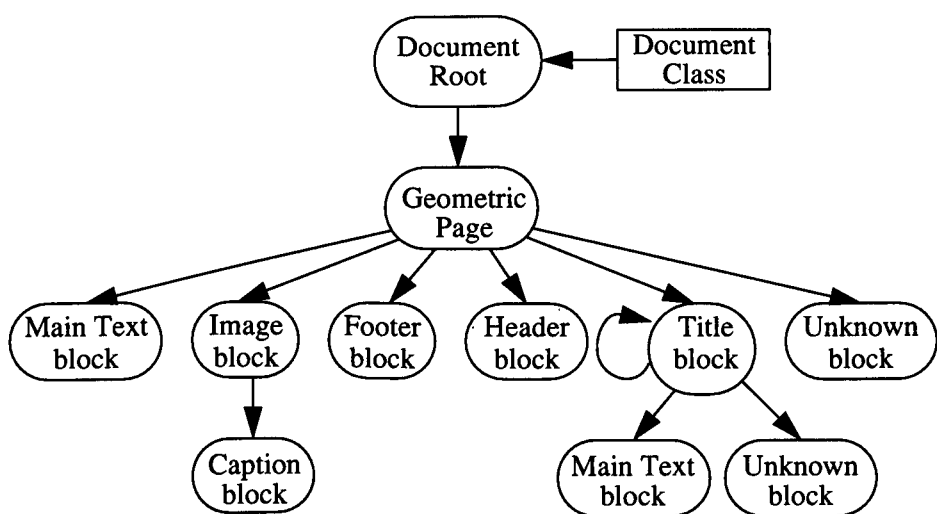


Figure 22: The STASIS document data structure



## **Final System Development and Engineering**

---

The structure knowledge source could attempt to make more intuitive links between blocks after the classification but it was decided that the system should only represent links between blocks which it was confident existed logically and not speculate on inter-block relationships. The resulting data structure may not represent much of the document's logical structure, but it will not pass on to any document understanding system inter-block links which are suspect or mis-leading

**This chapter has provided a detailed explanation of the development and internal workings of the final system. The output of this system is a high level description of the document from a mainly geometrical perspective. The inclusion of the advanced document analysis routines incorporates a basic level of document understanding into this output.**

# **Chapter 6, Analysis of System Results**

**There are three types of results produced by STASIS: results of the text-segmentation algorithm, results of the geometric block classification system and the results of the neural net document classifier. Detailed analysis of these results are included in this chapter, along with a thorough explanation and classification of the errors each sub-system produces and the context in which the errors occur.**

# 6.1 Introduction

The STASIS system can produce invalid results at various different stages of processing a document. The nature of these errors depends on a variety of factors and contexts. From a software engineering perspective there are certain exception cases in the text block segmentation routines which are not implemented. There are also weaknesses in the block tagging routines and column forming routines. From a theoretical perspective there are certain classes of documents which cannot be classified using the algorithms put forward in this thesis. This section identifies error generating contexts, explains the nature of the errors and discusses the various options available to rectify the invalid results. Juxtaposed with the discussion of error forming contexts are examples of STASIS' valid results in similar contexts.

# 6.2 Document analysis results

The algorithm used by STASIS to segment the text of a PDF document is a hybrid algorithm. Research into document analysis has suggested that hybrid strategies produce good results but rely on the presence of certain high level geometric features such as columns and text blocks (see section 3.2.2.4, "An assessment of Page Segmentation strategies" for a review of hybrid text-segmentation techniques).

There are aspects of the document analysis segmentation algorithm which have not been thoroughly addressed:

- the algorithm does not acknowledge the presence of graphical lines. This is an important issue which is required when segmenting form documents;
- there are rare, or special cases which the algorithm's general purpose routines do not handle well, for example, if a page contains text with an abnormally large leading then the algorithm may not form blocks well. Text typeset with Type 3 (bitmap)

## Analysis of System Results

---

fonts are ignored altogether by the algorithm; no font metric information is held by Type 3 fonts.

Even with these weaknesses the segmentation algorithm still produces acceptable enough results upon which to base further document processing stages. The algorithm produces optimal results when the text is well formatted and the paragraphs have a normal and consistent leading.

Whilst refining and modifying the segmentation algorithm and simultaneously engineering the blackboard architecture (which STASIS uses to tag text blocks), it became clear that the segmentation algorithm *should* have been incorporated into the blackboard framework. The blackboard framework can provide the flexibility to easily model all the exception cases which occur infrequently in the segmentation of PDF documents (and bitmap document images). Knowledge sources such as “word KS”, “line KS”, “leading KS” and “paragraph justification KS” could all combine to help produce valid text blocks. These knowledge sources would exist on a level of abstraction below that of the block tagging and page layout analysis knowledge sources.

STASIS’ segmentation process was not incorporated into the blackboard framework because this research focused on finding a method of classifying documents; the blackboard framework was developed for this purpose and the segmentation algorithm was always seen as a ‘data provider’ for the classification process. The segmentation algorithm used by STASIS was based upon the prototype’s segmentation algorithm developed at Adobe and this system was not blackboard-oriented.

There are specific error cases (as opposed to weaknesses) in the STASIS document analysis technique which are identified and discussed in section 6.3, “Document analysis errors”.

“Appendix I: STASIS Screen Shots” presents twenty example documents which have been segmented successfully by the STASIS system. Each screen shot has been saved as an image. Each image has been marked up with the tags of various blocks segmented by STASIS. Discussion notes accompany each image.

# 6.3 Document analysis errors

Two classes of invalid results are produced by STASIS' document analysis processes. In a sense the first class is not an invalid result but a failure of the system to analyse graphical elements. Figure 30, on page 199, shows a PDF document which has been segmented by the STASIS system. From this figure, it can be seen that the graphical diagram has not been recognised by STASIS as a logical entity. Furthermore, the text components of the logical diagram are not associated with that diagram but are considered separate text blocks.

The second class of invalid results arises from forming bad geometric text blocks. They are known as segmentation errors. Segmentation errors arise from one of four different contexts. A statistical analysis of the frequency of occurrence of these errors is provided at the end of this chapter in section 6.8, "A statistical analysis of STASIS".

## 6.3.1 Bad API lines

Creating geometric text blocks in the PDF model could be achieved by taking each word as a separate logical entity and constructing lines and then blocks using any one of the bottom-up algorithms described in the document analysis literature survey (section 3.2, "Literature Survey"). Yet each of these algorithms has a number of contexts in which they do not function well and so produce invalid results. STASIS uses the text lines that the Adobe API produced with its in-house line building functions. There are contexts in which the API text lines are invalid, however, STASIS does not check the validity of these lines. These contexts occur at an acceptably infrequent level for the purposes of the STASIS system. If STASIS was to check each line's validity it would dramatically increase the processing time of the segmentation routines.

Figure 23 shows part of a PDF document which has been decomposed by the STASIS system; the figure shows a page of a newspaper. STASIS has drawn boxes around the perimeter of the geometric blocks it has formed using its segmentation algorithm.

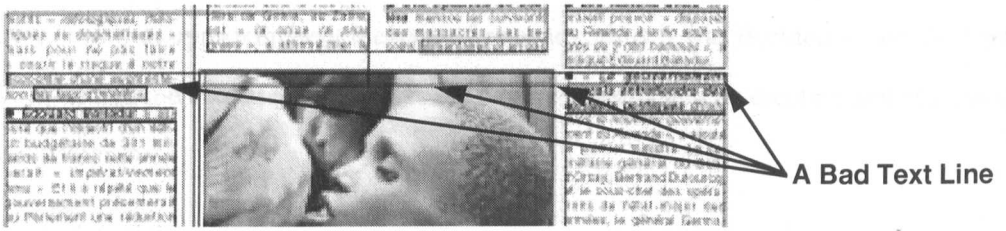


Figure 23: A page portion of 'Le Figaro' showing a single API text line definition error.

Figure 23 is an example of an API line definition error. At the top of the central image (showing part of a child's head) there is a single geometric block which is one text line in height. This block contains two separate text lines which are formatted inside the columns either side of the central picture. The API line finding algorithm has misjudged the layout of the page and wrongly grouped both of these lines into one line. API type errors rarely occur in documents in which the page layout is simple.

### 6.3.2 Assigning poor line attributes

One of the first processing techniques that the segmentation routine uses to create text blocks is to separate all the lines into lists based on a style identification key formed from the line's font and point size attributes. Each line is given a font and point size key by analysing one word from the line; the line is given the point size and font attributes of that word. That word is chosen by taking the total number of words in the line and dividing the total by three. The result is the index (in the line) of the word which will dictate the geometric properties of the line. The problem with this approach is that the chosen word may not accurately represent the font and point size of the majority of the words in the line, or in the geometric block in which the line resides.

Previous attempts to rectify this algorithm's shortcomings included building up a histogram of the word styles in the line and picking the most frequently occurring style. However, this technique also had drawbacks. Consider a line in which three quarters of the line is set in an italic font, the remainder of the line is set in a plain font and the geometric block in which the line resides is set in a plain font. This particular line would

## Analysis of System Results

---

be given the font attributes of the italic font instead of the plain font. Both classes of errors occur at approximately the same frequency; thus it was decided to use the '3rd word' technique because of the increased overhead generated by creating and analysing the histogram.

When a line is assigned poor geometric attributes over segmentation occurs. The true geometric block is typically over segmented into three portions: the top section of the block; a section containing the single line with the different attributes; and the bottom section of the block. A specific block merging operation could be developed which would merge together the geometric blocks involved in this error context, but the heuristics involved in developing such an algorithm may produce invalid text blocks in other contexts. Generally, over segmentation occurs when the word which represents the style key of the text line is formatted in a bold or italic font style. Subsequent processing by the advanced level of document analysis will make this single line block a title block because of the increased prominence the font of the block has over the text styles of the blocks beneath it. A more serious repercussion would arise if the system merged a real title block into a text block. This could occur in other contexts as the system attempted to rebuild over segmented blocks.

Figure 24 illustrates the consequences of assigning poor geometric attributes to a text line. The figure shows a portion of a page in which three blocks are present. The central block has been assigned the geometric attributes of the third word in its one and only member line. Consequently, it has been segmented out of the geometric block in which it *should* be placed and is considered by the system to be a more prominent block than the text blocks surrounding it.

### 6.3.3 Bad geometric block forming

There are certain contexts in which the STASIS block forming algorithms will misinterpret their input and produce invalid output even when they have been presented with *bona fide* lines by the Acrobat Exchange API line building routines. These contexts

through the initial notes setup using the new user ID. Add an icon for the user's old mail database and copy all of the documents and paste them into their new mail database. All of these documents will appear as unread. You can easily change this unread state by selecting Tools | Unread Marks | Mark All Read. Add any additional local databases and test the automatic mail forwarding by sending the user's corporate identity an e-mail message. This message should be sent to your domain and delivered to the correct user. If this test fails, check the forwarding address in the old user document. A common mistake with forwarding addresses is usually misspelled

A Bad Text Line

*Figure 24: A text line with invalid geometric attributes*

are rare and have little effect on the document classification process, but they should be acknowledged. Figure 25 and Figure 26 both show instances of this problem context.

The symptoms of the problem can be seen in both diagrams: the creation of small geometric blocks inside a larger geometric block. In actuality, the smaller blocks are not present 'inside' the larger block. The larger block has been created by merging various blocks together following the rules set out in section 5.3, "Final System Document Analysis of PDF". The presence of the individual smaller blocks indicates that they were rejected by the merging algorithm. The primary cause of the rejection is because of their geometric position on the page. For all the individual smaller blocks shown in Figure 25, there is a degree of incompatibility with the geometric parameters of the larger block.

In Figure 25 the problem paragraph is left justified. The column it resides in is small in width and the style of the paragraph dictates that the first line of the paragraph must be indented. Furthermore, because the column width is small, there is a high percentage of hyphenated words. The API word finding routines are good at locating the 'second' half of logical words which have been separated by line breaks. STASIS does not check to see if logical words have fractured bounding boxes. Consequently, the bounding box of the line on which the second half of the hyphenated word resides starts with the start of the *next* logical word *after* the finishing hyphenated word. These factors all contribute to a large degree of variance in the geometric positions of the bounding boxes of the individual lines inside one logical paragraph. The STASIS system is reluctant to check for fractured bounding boxes within a single logical word because of the overhead



## Analysis of System Results

required to do so for each word in a document.

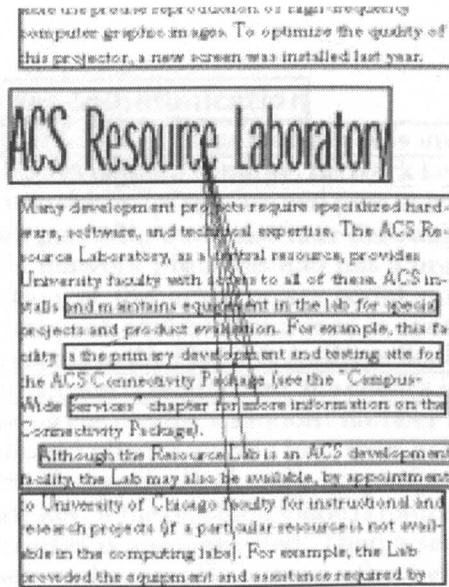


Figure 25: Bad block forming example 'A'

Figure 26 illustrates the same problem as that illustrated in Figure 25, but which arises from a slightly different context. In this specific case, three blocks have been segmented from the text, another larger block (containing three lines) overlaps them. The three lines were not granted membership into the other blocks during the segmentation routines, as they each have an indent. The indent was enough to give the segmentation routines reasonable doubt regarding their claim to membership of the other blocks. Later in the block forming process the three lines were merged into one block. This can happen because the vertical distance between the three lines is constant. This has confused the merging algorithm. It has mistakenly thought that the constant vertical distance is a *bona fide* block inter-line value.

This is a very dangerous error context which can occur to candidate title blocks, if they have been formatted at a constant distance from one another and lie approximately in line with each other on the page. STASIS applies a heuristic which attempts to prevent the creation of illegal blocks in this manner. The heuristic makes sure that the leading value of any block is never more than a set limit above the point size of the text in that

# Analysis of System Results

block. All heuristics have weaknesses and there are certain contexts in which this heuristic cannot identify illegal blocks as they are created.

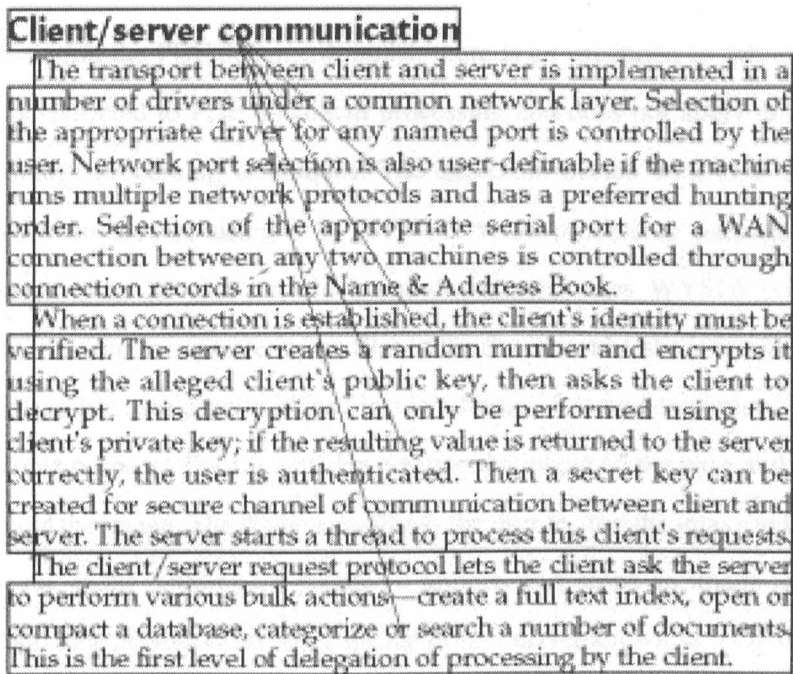


Figure 26: Bad block forming example 'B'

Adjusting the existing block forming routines to prevent the creation of invalid blocks in these contexts could be a poor software engineering decision for a three reasons.

Firstly, these error cases are infrequent. In the majority of instances the combination of factors required to create these error forming contexts rarely occur. Adjusting the existing algorithm may have an adverse effect on the majority of contexts in which the existing algorithm works perfectly.

Secondly, the computational overhead would be significantly increased in all cases, not just the error forming contexts.

Thirdly, there is an easy method for detecting the presence of bad blocks. The invalid geometric blocks shown in Figure 25 and Figure 26 can be recognised by examining their bounding boxes. If there is an overlap, then an error has occurred in the block forming routines. This simple check can be abstracted into the following axiom:

## Analysis of System Results

---

**there should be no overlap between any of the bounding boxes of segmented image regions created by a document analysis segmentation and decomposition algorithm.**

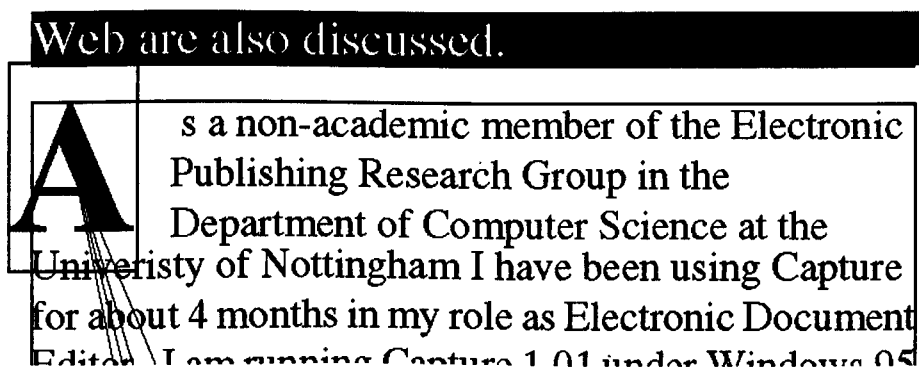
In practice, within the PDF document model, there is an exception to this axiom. The exception is restricted to PDF document processing and does not apply to traditional document image processing techniques. The exception occurs with PDF documents which have been created using a WYSIWYG word processing package (such as Framemaker or QuarkXPress) and which include bitmap images. WYSIWYG packages allow the user to import separate image files in the document he/she is working on. The imported image files are usually placed inside a frame of some description. The frame serves a double purpose. The frame helps position the image on the page and allow text to flow around the image. The frame also allows the user to crop the image to the desired size. However, if the author of a document imports and crops an image into his/her document and then prints to PDF via a special printer driver known as 'PDFWriter', the resulting PDF file contains the entire bitmap image and not just the area of the image visible to the reader. When the document is viewed, all one sees is the cropped image (or visible bounding box), but when one extracts that image from the PDF document model the entire original image is retained. A direct consequence of this shortcoming of PDFWriter is that the actual bounding box of the image resource is not the same as the visible bounding box. Typically the actual bounding box will extend over the image's visible bounding box and start to overlap with the bounding boxes of surrounding geometric blocks.

Using the axiom stated previously and the knowledge that the bounding boxes of images do not contribute to the creation of the type of error illustrated in Figure 26, one can simply locate bad blocks by checking for overlaps amongst text blocks. Once an overlap has been detected, one should carefully examine the context of the problem and check the geometric parameters of the invalid geometric blocks and the attributes of their text lines. This solution is a hypothetical suggestion which has not been implemented in the

STASIS system. No implementation of this solution is provided because of a lack of implementation time coupled with a low priority weighting assigned to this problem by the author.

### 6.3.4 Dropped Caps

STASIS does not seek out dropped caps in documents. It was felt that this would be paying too much attention to an insignificant detail. In retrospect they should have been looked for as an exceptional case of paragraph formatting. Recognising multiple instances of dropped caps in a single document could help document understanding algorithms identify logical entities within a document. A more immediate advantage of recognising dropped caps would be the ability to stop the dropped caps dominating the text blocks around it. Dropped caps are only intended to draw the reader's eye. They have no logical role in the documents structure. Yet, by not formally acknowledging dropped caps, the STASIS system lets them form geometric blocks of their own which are then processed normally by the system. Figure 27 shows the implications of not recognising the dropped cap as a geometric feature of a logical paragraph: inappropriate links to other blocks and a divorce from its true logical paragraph.



*Figure 27: A paragraph formatted with a dropped cap style*

# 6.4 Advanced analysis results

The advanced analysis of PDF documents performed by STASIS is extremely successful. Advanced analysis can be defined as the recognition and classification of low level document elements and the recognition of low level logical dependencies between these elements. Only in form documents did the block classification process produce poor results. This is partly because form documents have their logical structure explicitly defined by their layout. The logical components of form documents are defined by their exact position in the form, for example, logical column titles are found at the top of columns. Another reason is that STASIS does not process graphic lines. Graphic lines are an extremely important source of geometric information in form documents.

The examples in Appendix I illustrate the results of the tagging process. There are accompanying notes with the examples which discuss the relevant issues. The error forming contexts which occur in the process of advanced analysis are discussed in section 6.5, "Advanced analysis errors".

The algorithms which the knowledge sources use to gather information and then infer from that information are surprisingly simple. Many are simply statistical, others find patterns amongst the data they see by applying basic pattern matching techniques. The consistency of recurring patterns is an important feature that many knowledge sources rely on and yet it is so simple to check for.

One of the major advantages of the blackboard framework is that basic algorithms can be applied (when they are needed) and their results combined hierarchically to produce a powerful system, for example, the text frequency knowledge source 'guesses' at the style of the document's main text style by statistically processing the text styles and their frequency of occurrence in the document. This is very simple and yet very successful. Similarly, the image knowledge source simply finds the positions of all the images in the document. The column knowledge source then uses both these sources of

information (through the medium of the blackboard) to find a vast amount of information about the document that is vital to the document classification process.

# 6.5 Advanced analysis errors

The STASIS system was originally designed to be able to process documents from any logical class. This is an extremely ambitious goal. The problem is compounded by the fact that the geometric styles amongst documents of the *same* logical class varies immensely. The wide range of input documents forces the STASIS system to base its tagging and feature finding algorithms on generalised page layout rules. By making the algorithms general, STASIS achieves a good level of analysis in all classes of documents it processes.

The disadvantage of being a general system is that STASIS cannot apply document-specific recognition and classification routines to the documents it processes. Consequently, a variety of invalid results are generated within the advanced analysis stage of document processing in certain contexts. The nature of the errors vary from tagging text blocks with incorrect tags to misinterpreting the logical role of a block within a document.

## 6.5.1 Linking errors

The least significant of these errors occurs when a text block is made into a logical title based on the evidence provided by a poorly formed logical link to another text block. The purpose of the link is to provide some information to the system on the nature of the logical role of the block in the document. Linking errors generally occur in documents which have a complex page layout, for example, a newspaper. Ideally a title block should link to the body of text which it logically relates to.

## Analysis of System Results

---

Newspapers have a large number of title blocks and the majority of these blocks are correctly processed by STASIS. When an invalid title block is created (in a complex document such as a newspaper) there is no significant effect on the features extracted from the document in the classification process. An invalid title would have a more profound effect on the document features of an academic document in which there are few title blocks. However, because academic documents generally have simple page layouts, linking errors in this class of document are extremely infrequent.

### 6.5.2 Column Forming errors

The Column KS algorithm which creates column bounding boxes on a page of a PDF document is entirely dependent upon the validity of the decisions made by the Text Frequency Knowledge source. The Column Knowledge Source only looks at the main text blocks on a page. It is realistic to expect a page in a complex document such as a newspaper to contain no blocks which are deemed to be main text blocks, for example, the pages containing the classified advertisements. The column finding algorithm would produce poor results when presented with such a page. This problem is generally found in newspaper type documents only.

The document features which describe the attributes of the columns in a document are designed in such a way that the effects of one poorly processed page are minimal. The 'maximum number of columns on a page' feature is unaffected by a single poorly processed page. In a newspaper document this value will still be large and typically greater than three. However, the feature which describes the 'consistency of column styles throughout the document' will be affected. Even so, for a newspaper document this has a minimal effect. One of the characteristics of a newspaper is that the column style does vary dramatically from page to page. The Column Knowledge Source will, in the worst case, recognise one or no columns on a page. This will be inconsistent with the results of the column detecting routines on other pages and the 'consistent column style' feature will be set to false. This is the expected result for a newspaper type document.

When this particular error making context occurs in a non-newspaper type document then the consequences can be more serious. In trials held with the STASIS system it was found that sometimes the Column Knowledge Source poorly diagnosed an academic document as having an inconsistent column style. This invalid feature was not enough to cause a invalid document classification, but it did reduce the 'clarity' of the decision.

### 6.5.3 Blackboard diagnosis errors

The STASIS system is weak at tagging blocks which are formatted in certain styles. However, these styles occur infrequently and when they are present they have little effect on the overall document classification result.

#### 6.5.3.1 Hanging headers

All title blocks are formed in the STASIS system by analysing their *umbrella of influence* over the text blocks which are formatted beneath them on the page. One particular style of title does not extend an umbrella of influence over text blocks in the vertical orientation: 'hanging headers'. Hanging headers is the phrase used to describe text blocks which have a logical title capacity within the document but are formatted to one side of the text blocks they are associated with. They hang off the text blocks. This escapes the title detection routines used to help classify title blocks. This formatting style is difficult to detect. If the system were to look horizontally from a text block to check for possible logical associations, then documents in which hanging headers were present would be processed correctly, but other documents would be processed incorrectly. As hanging headers occur relatively infrequently in comparison with traditional title blocks formats, it is inefficient to adopt a horizontal association search algorithm.



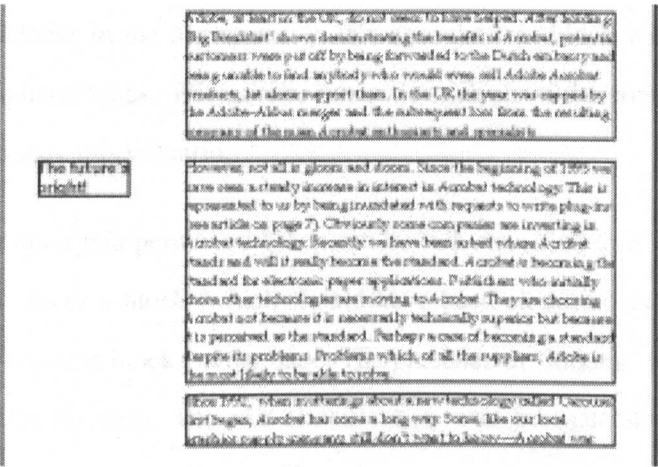


Figure 28: A document formatted with 'hanging headers'

Documents in which hanging headers occur are almost always documents with an academic structure and never newspapers. The result of not associating a hanging header with its text block is that the hanging header will not be classified as a title block. Instead it will be tagged 'unknown'. STASIS will, therefore, not register the style of the hanging title as a title style and the list of title styles that STASIS compiles during the processing will be deficient. This does not have a detrimental effect upon the document classification process as academic documents typically only have a few title styles compared with newspaper documents. Figure 28 illustrates a document formatted with hanging headers.

6.5.3.2 Captions of diagrams

Diagrams and pictures which are drawn with graphic elements are invisible to STASIS because of the limited graphical analysis implemented in the system. Consequently, the *bona fide* captions to these diagrams are difficult to detect as captions.

6.5.3.3 Peripheral entities

The STASIS system detects headers and footers based upon rules inferred from the repetitive occurrence of blocks which are formatted in the same text style and set in the same area of the page over a series of pages in a document. Rarely, a text block which

# Analysis of System Results

is logically *not* a peripheral block will exhibit the *same* geometric properties as the *bona fide* peripheral blocks in the document. In this case the text block will be incorrectly tagged as a peripheral block. Without the semantic analysis of the content of a block it is difficult to remedy this scenario.

Alternatively, a *bona fide* peripheral block can be mis-classified as a block of another class. Figure 29 shows a block (which should have been classified as a header block) linking into various text blocks in the page. The presence of 'linking' indicates a badly judged block classification. The mis-classification was brought about because not enough instances of this type of header block were present in the document. The Header Knowledge Source did not have enough information to generate, or infer, a specific knowledge rule for this type of header block.

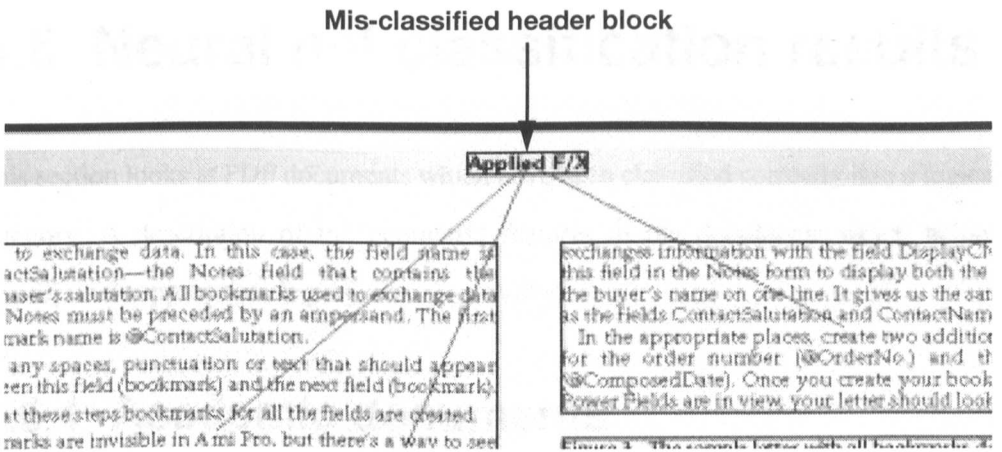


Figure 29: A document portion with a mis-classified header block

Figure 30 shows a footer block which has been poorly classified as a block of unknown type. Consequently, a text block present in the diagram has inappropriately linked to the misleading footer block. STASIS mis-classified this block because there was not enough information present to infer a rule. In this instance, the document was only one page long. This makes it extremely difficult for the STASIS system to generate rules based on inference by example.

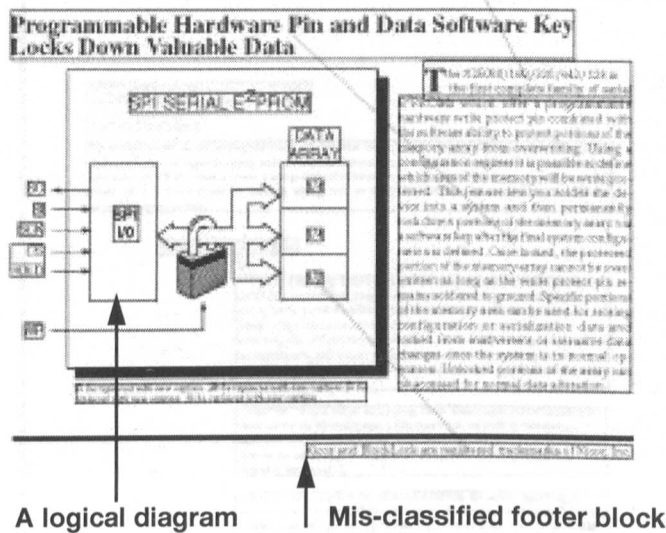


Figure 30: A document portion showing the a mis-classified footer block

# 6.6 Neural net classification results

This section looks at PDF documents which have been classified correctly into a logical category. A description of the geometric features of the documents which helped identify their appropriate logical class is provided.

## 6.6.1 Academic documents

Figure 31 shows a page of a PDF document which STASIS has classified as a logical academic document. Some of the features which the STASIS system detected cannot be seen from a single page image. Table 17 shows the full report generated by STASIS for this document. Many of the features stated in the report will not appear significant until they are compared and contrasted with the reports created by processing other documents.

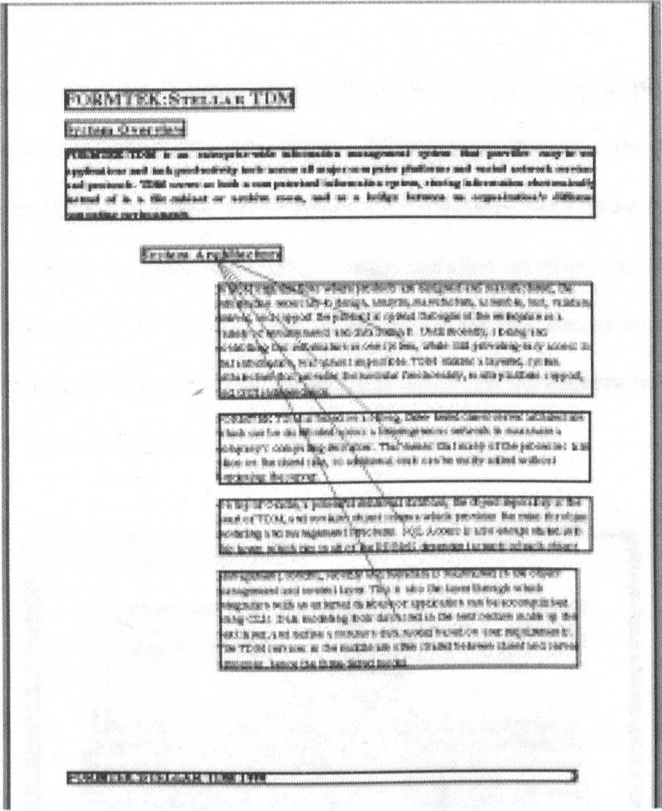


Figure 31: An academic document, STELLA.PDF

|   |
|---|
| *****   |
| STASIS system report for document titled:                             |
| /C/USR/WSL/MYDOCU-1/NEURAL-1/TRAIN/ACADEMIC/STELLA.PDF                |
| *****   |
| There are 14 text styles of which 1 are deemed to be main text styles |
| Footers detected  |
| There are footers present on 97.14% of the pages                      |
| There are 8 title styles  |
| A Super Title has been detected                                       |
| Results of graphics analysis  |
| Total paths detected: 140   |
| Total straight paths detected: 140                                    |
| Largest number of straight paths on one page: 137                     |
| Average number of straight lines per page: 4.00                       |
| ColumnStyle Inconsistent throughout document                          |
| Image consistency   |
| There are 19 images on 35 pages, which gives a ratio of 0.542857      |
| NEWSPAPER == 0.0005   |
| FORM == 0.0009  |
| BROCHURE == 0.0022  |
| ACADEMIC == 0.9725  |

Table 17: The STASIS report for the STELLA.PDF document

The following features are of particular interest. There are under twenty text styles in the entire document of which eight are candidate title styles. There are footers present on the majority of pages. There is a relatively low average number of images per page and there were no captions detected for these images.

# 6.6.2 Newspaper documents

Figure 32 shows a processed page taken from a Spanish newspaper and Table 18 presents the report data for this document. Newspaper documents are much more geometrically complex than many other types of document; they have complex column layouts, a large number of text styles and a large number of title text styles. They also contain a high frequency of images and graphics. STASIS has detected all of these features in this example. The STASIS system has detected no footers in this newspaper, but this anomaly has not affected the overall classification.



Figure 32: A newspaper document, SPAIN0.PDF

|   |
|---|
| +++++   |
| STASIS system report for document titled:                             |
| /C:/USR/WSL/MYDOCU-1/NEURAL-1/TRAIN/NEWS/SPAIN0.PDF                   |
| +++++   |
| There are 107 text styles of which 1 are deemed to be main textstyles |
| Headers detected  |
| There are headers present on all pages                                |
| Captions have been detected   |
| 78.787880% of the captions are in a similar position: left of         |
| All Captions have the same font metrics                               |
| There are 73 title styles   |
| A Super Title has been detected                                       |
| Results of graphics analysis  |
| Total paths detected: 5963  |
| Total straight paths detected: 4670                                   |
| Largest number of straight paths on one page: 1188                    |
| Average number of straight lines per page: 89.81                      |
| ColumnStyle Inconsistent throughout document                          |
| No image consistency  |
| There are 132 images on 52 pages, which gives a ratio of 2.538461     |
| NEWSPAPER == 0.9881   |
| FORM == 0.0006  |
| BROCHURE == 0.0180  |
| ACADEMIC == 0.0089  |

Table 18: The STASIS report for the document SPAIN0.PDF

6.6.3 Brochure documents

Brochure documents are, for the purposes of this system, a document containing one logical article. The class of brochure documents covers a wide range of geometric page variations. Figure 35 illustrates a typical brochure which describes an artist’s use of Adobe Illustrator™ – a software program which specialises in creating graphic illustrations. In heuristic terms, the brochure is ‘tidier’ than a newspaper, yet still more geometrically diverse than an academic document. Translating this heuristic knowledge into computable knowledge is difficult. Table 19 presents the extracted knowledge for the brochure document presented in Figure 35.

STASIS acknowledges the presence of a more rigid structure than that present in the newspaper example, through the detection of several document features. STASIS infers that the column style is consistent throughout the document. There are less text styles present. In fact, the number of text styles is closer to that of an average academic document than that of a newspaper. There are captions present for some of the images, and those captions are consistent in their text style and their position relative to their image.

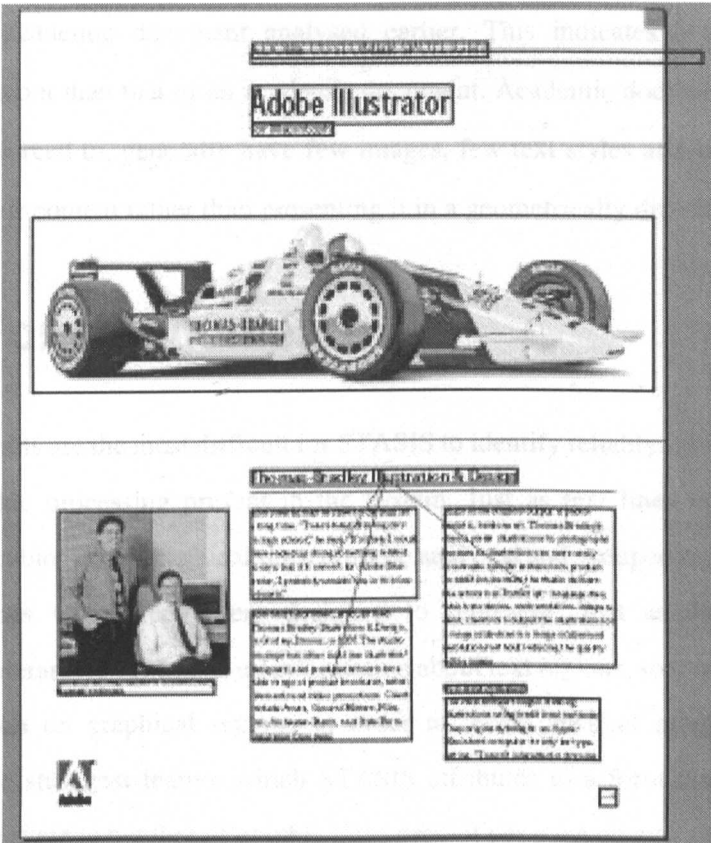


Figure 33: A brochure document, THOMBRAD.PDF

|   |
|---|
| +++++   |
| STASIS system report for document titled:                             |
| /C:/USR/WSL/MYDOCU-1/NEURAL-1/SGMLDE-1/THOMBRAD.PDF                   |
| +++++   |
| There are 12 text styles of which 1 are deemed to be main text styles |
| Headers detected  |
| Header styles are consistent  |
| There are headers present on 50.00% of the pages                      |
| Footers detected  |
| There are footers present on all pages                                |
| Captions have been detected   |
| All Captions are in the same position relative to their image: below  |
| All Captions have the same font metrics                               |
| There are 7 title styles  |
| A Super Title has been detected                                       |
| Results of graphics analysis  |
| Total paths detected: 37  |
| Total straight paths detected: 31                                     |
| Largest number of straight paths on one page: 21                      |
| Average number of straight lines per page: 15.50                      |
| Column Style Consistent throughout document                           |
| No image consistency  |
| There are 4 images on 2 pages, which gives a ratio of 2.000000        |
| NEWSPAPER == 0.0000   |
| FORM == 0.0001  |
| BROCHURE == 0.9984  |
| ACADEMIC == 0.0010  |

Table 19: The STASIS report for THOMBRAD.PDF

However, there are more images per page and more graphics per page in the brochure than in the academic document analysed earlier. This indicates a slightly more ‘colourful’ layout than that of an academic document. Academic documents, although they are not forced to, generally have few images, few text styles and concentrate on delivering their content rather than presenting it in a geometrically diverse layout.

### 6.6.4 Form documents

Form documents are the most difficult for STASIS to identify reliably. This is due to the lack of graphic processing present in the system. Just as text lines are joined into geometric text blocks, so graphical lines must be analysed and grouped together into the abstract shapes which they were designed to represent. Just as the blackboard architecture hierarchically builds up information about text regions, so it must also build up information on graphical regions in order to locate abstract graphical entities. Currently, the strongest feature which STASIS attributes to a form document being present is the average number of graphic lines present per page on average. The results of the classification technique have proven that this is *not* enough to unambiguously identify form documents.

Figure 34 shows a page from a form document. Table 20 lists the features extracted by STASIS during the processing of that document. It can be seen that there is a high number of straight lines present in this document, but not as much as in the newspaper example presented earlier. Ignoring the graphical features, the report that STASIS generated for the form document resembles the report for the academic document. Notably, there are few text styles, headers and footers are present and there are no images. This similarity manifests itself in the classification process as the academic node is the ‘next closest’ in terms of ‘node weight’ after the form node.



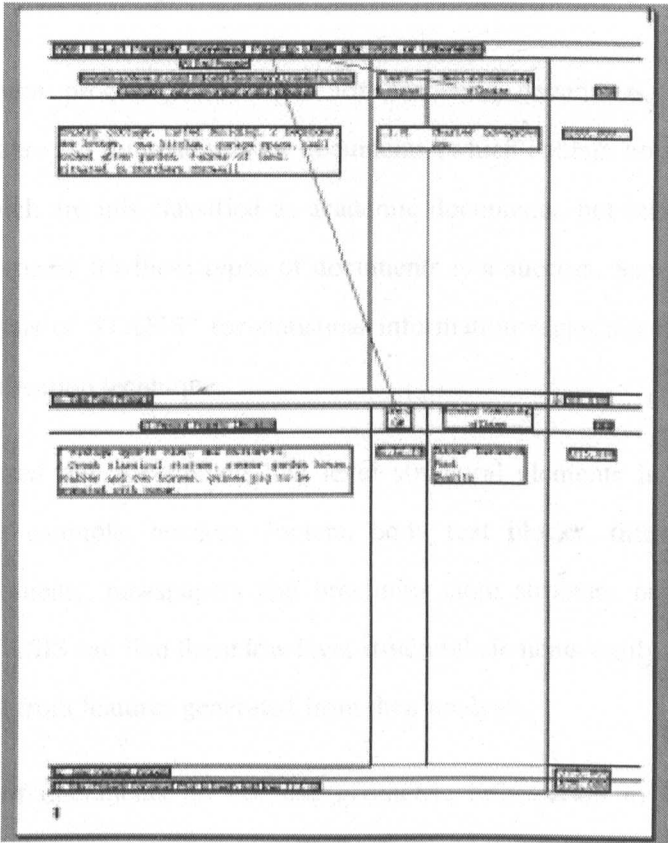


Figure 34: A form document, TAX11.PDF

|   |
|---|
| +++++   |
| STASIS system report for document titled:                             |
| /C/USR/WSL/MYDOCU-1/NEURAL-1/SGMLDE-1/TAX11.PDF                       |
| +++++   |
| There are 11 text styles of which 2 are deemed to be main text styles |
| Headers detected  |
| There are headers present on all pages                                |
| Footers detected  |
| There are footers present on all pages                                |
| There are 8 title styles  |
| A Super Title has been detected                                       |
| Results of graphics analysis  |
| Total paths detected: 159   |
| Total straight paths detected: 159                                    |
| Largest number of straight paths on one page: 98                      |
| Average number of straight lines per page: 31.80                      |
| ColumnStyle Inconsistent throughout document                          |
| NEWSPAPER == 0.2070   |
| FORM == 0.6247  |
| BROCHURE == 0.0052  |
| ACADEMIC == 0.3552  |

Table 20: The STASIS report for TAX11.PDF

### 6.6.5 Summary

The classification process works well for academic documents, brochures and newspapers. There are some newsletter documents (which contain no images and few text styles) which are mis-classified as academic documents, but other than this the classification process for these types of documents is a success. See section 6.8, “A statistical analysis of STASIS” for statistical information regarding the error rate of STASIS’ classification technique.

STASIS is biased towards finding low level structural elements in these types of documents, for example, headers, footers, body text blocks, titles and captions. Academic documents, newspapers and brochures store structure *implicitly* in their appearance. STASIS can find these low level structural elements easily and classify the documents well from features generated from their analysis.

Certain types of documents do not use geometric information to implicitly store structure. These documents are not classified efficiently by STASIS, for example, form documents. Form documents use geometric information (in the form of graphic lines) and semantic content to *explicitly* define the logical role of their composite logical entities. Column headers, row headers and cell contents are logically defined by their exact position in the form and their semantic content. Typically, there are few low level logical entities in form documents.

Early on in STASIS’ development it was hoped that the system would be able to classify business letter documents. Unfortunately there were no features (or groups of features) with which STASIS could identify a business letter with. Business letters, like forms, do not contain many low level elements. They have body blocks which STASIS could identify and tag, but other than that, they consist of letter-specific entities such as ‘sender address’, ‘sign-off’ and ‘salutations’. Without using model-driven letter document-specific recognition routines it was impossible to classify letter documents.

# 6.7 Neural net classification errors

There are certain classes of documents which exhibit the geometric properties of documents which are not of their logical class.

Figure 35 shows a logical brochure document which initially had been mis-classified as a newspaper document. This document was processed by the system early in its development. The neural net was still relatively ‘inexperienced’ and required more examples of brochure documents to establish which document features (or combination of features) could be used to unambiguously identify brochure documents. The document illustrated in Figure 35 was ideally suited to train the neural net with, because it displayed all the required geometric features of a brochure document (a large number of type faces, a complex page layout, a fairly short document length and an average distribution of images) without displaying many of the features which are present in newspaper documents (a very large number of type faces, an extremely complex page layout and a high frequency of images).

Newsletters documents are sometimes classified as brochures. Ideally, a newsletter should be associated with the logical class of newspapers. They have the same logical structure which comprises mainly of logical articles and sub-stories. However, they also have geometric properties which are closely related to brochures, for example, newsletters are less than ten pages in length, they have a consistent column format and contain fewer text styles than broadsheet newspapers. Furthermore, newsletters may have a smaller average number of images per page than a broadsheet newspaper. All these geometric attributes are closely related to the geometric attributes of brochures. The inverse of this scenario also exists; some brochures exhibit the geometric properties of newsletters.

The layouts of some brochure documents are closely related to the layouts of some academic documents. The possible page layout of a brochure is wide ranging. It may be either rigidly formatted in only a few fonts and contain uniform headers and footers or

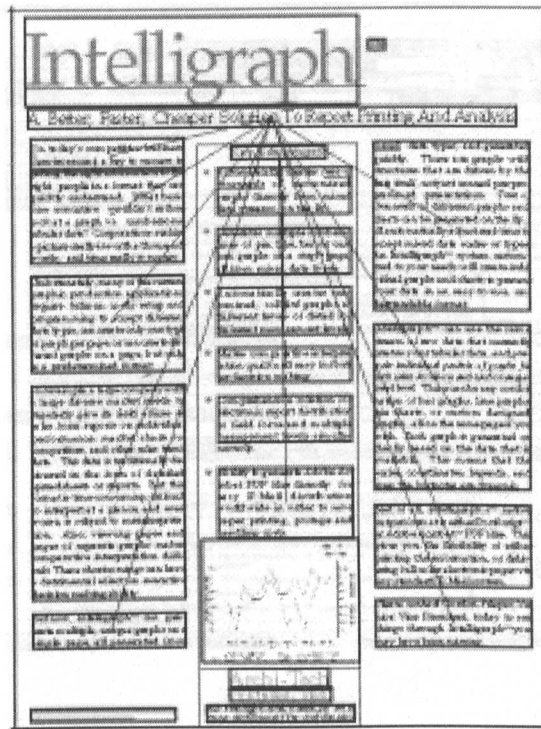


Figure 35: A page of a logical brochure

complexly formatted with a wide variety of fonts, a large frequency of images and irregular column formats with no headers and footers. Generally the layout of a brochure lies somewhere in between these extremes. When a brochure does not exhibit these ‘extreme’ properties, STASIS can identify it as a logical brochure accurately.

Pages from documents which display the geometric attributes of more than one type of logical document (which have consequently confused the neural net) are displayed in Figure 36 and Figure 37. These pages are taken from a logical form document which is four pages in length. The first two pages of the document display the typical geometrical and logical characteristics of a form document (Figure 36 shows the first page of this document). The third and fourth pages of the document display the typical geometrical and logical characteristics of either a brochure or an academic document (Figure 37 displays the fourth page of this document).

0040

Driver Connection Exception Statement for Drivers

95

Driver Information

Vehicle Information

Driver License Information

Driver History

Driver's Current Status

Figure 36: The front page of a logical form document

In the classification process, the STASIS system declared the document to be a member of the brochure class. However, the decision made by the neural net was ambiguous; the weight of the node representing the brochure class of documents was not significantly greater than the weight of the nodes representing the academic and form document classes.

The difficulty of classifying a document which contains pages which display attributes of distinctly different classes raises an interesting area of research. For a document of the type illustrated in Figure 36 and Figure 37, it would be optimal to apply specific document understanding routines to specific pages. In other words, a document processing system should apply a form document understanding algorithm to the page shown in Figure 36, and a brochure document understanding algorithm to the page shown in Figure 37.

Figure 37: The back page of a logical form document

The results which the STASIS system has produced have indicated that it is possible to make inferences on the nature of individual page layouts. It should be possible to recognise the presence of table and form structures on a page if they represent a significant portion of the page. This would not necessarily obstruct the global goal of document class recognition. One could incorporate a flexible approach to logical document class recognition and allow a document of one logical class (for example an academic document) to have pages present in it which can be logically described as belonging to another class (for example a table or a form or an advert). Currently, STASIS does not attempt to classify individual pages.

### 6.7.1 Neural Net shortcomings

The neural net classifier delivered all it promised; it learnt from examples and detected patterns amongst the document features it saw. Judging by the statistical analysis of the classification results (see section 6.8, “A statistical analysis of STASIS”) it can be seen that the application of a neural net classifier was a success.

However, nearly all the documents which were ‘classified’ with the neural net were *bona fide* members of one of the four target classes of document. Letter documents, for example, were rarely processed by STASIS. On the occasions when letters were processed, the classification technique declared that the letters were brochure documents. This was not because the letter document resembled a brochure but because the neural net forced itself to make a classification and the target node for brochure documents was the closest ‘match’ it could find. From this experimentation it can be said that the neural net classifier is weak at detecting the presence of documents which are *not* a member of *any* of the target output classes.

Another negative aspect of using a neural net classification system arises when one analyses an output (or target) node’s weight after a classification has been performed. There is a tendency to relate the ‘weight’ of the node to a ‘confidence’ which the neural net has in its classification, for example, if node A has a weight of 0.897 and node B has a weight of 0.009, then one could be forgiven for associating the magnitude of the weights with a magnitude of confidence in a classification decision. This assumption does not become a problem until one wishes to look at the second and third ‘choices’ of the neural net classifier. In such a situation, it is unreasonable to declare that the weighting of a target node represents the neural net’s confidence in that node being the desired target node of the classification process.

## Analysis of System Results

---

Typically, the output of a neural net classifier will be biased towards the node which has the highest weight (this node will be referred to as the ‘primary node’, the node with the next highest weighting will be referred to as the ‘secondary node’ and so on). The magnitude of the difference between the primary node and the secondary node should be acknowledged as a magnitude of the degree to which the neural net ‘recognises’ a pattern and not as a magnitude of the degree to which the secondary node represents the logical target class. From these observations, it is clear that neural nets are not accurate at representing the degree to which the ‘next best’ node matches the profile of the input vector.

The neural net classifier has proven that it is possible to classify a document from meaningful features extracted from the document. Yet, the shortcomings of the neural net technique (which have been outlined in this section) could have important ramifications in a universal document processing system which should be designed to process all classes of document. Such a system may wish to inspect the second choice of a classification technique, particularly if there is a real similarity between the first and second choices, for example, a newsletter and a brochure. The classification process should provide a meaningful evaluation of the correlation between the document being processed, all the target classes and the class of ‘non-classifiable’ documents.

## 6.8 A statistical analysis of STASIS

Over three hundred documents were collected, primarily from internet resource sites, and tested with the STASIS system (see section 5.6.2.4, “Development of the STASIS neural net classifier” for more details). Three classes of errors have been identified: API errors (those created by the Acrobat API), segmentation errors (created by STASIS when decomposing a page) and document classification errors. This section presents and discusses a statistical analysis of the frequency of these errors.



## Analysis of System Results

The statistics representing API errors, segmentation errors and block tagging errors were partly compiled from human judgement; the author examined the documents that STASIS processed in order to identify these classes of errors.

The statistics concerning the recognition of a document's class were compiled by comparing the logical class indicated by STASIS to the true logical class of the document. This set of statistics is the most important of those compiled.

Table 21 lists the statistical information gathered from these tests. Figure 38 displays these results as bar charts.

| Class of Document | Total | API Errors (%) | Seg. Errors (%) | Tag Errors (%) | Class Errors (%) |
|-------------------|-------|----------------|-----------------|----------------|------------------|
| Newspaper         | 45    | 4 (8.9%)       | 7 (15.5%)       | 2 (4.4%)       | 5 (11.1%)        |
| Academic          | 136   | 4 (2.9%)       | 18 (13.2%)      | 9 (6.6%)       | 8 (5.8%)         |
| Brochure          | 77    | 5 (6.4%)       | 12 (15.5%)      | 2 (2.5%)       | 4 (5.1%)         |
| Form              | 53    | 4 (7.5%)       | 12 (22.6%)      | 4 (7.5%)       | 11 (20.7%)       |
| Total             | 311   | 17 (5.4%)      | 49 (15.7%)      | 17 (5.4%)      | 28 (9%)          |

Table 21: STASIS document analysis and classification statistics

### 6.8.1 Evaluating the statistics

The data which makes up the bar chart showing the relative API errors was compiled by identifying text lines which had been badly judged by the Acrobat API. If an invalid line was present in the document then its presence was recorded. The chart shows that the likelihood of an API error occurring is proportional to the complexity of the page layout. Newspapers have the most complex layout of all; thus they have the highest number of API errors on average. Form documents have an unusually high number of API errors because each time an entire text line crossed a vertical graphic line, an invalid text line was produced. Presumably the Acrobat API, like STASIS, does not take into account graphic lines when creating the text lines.

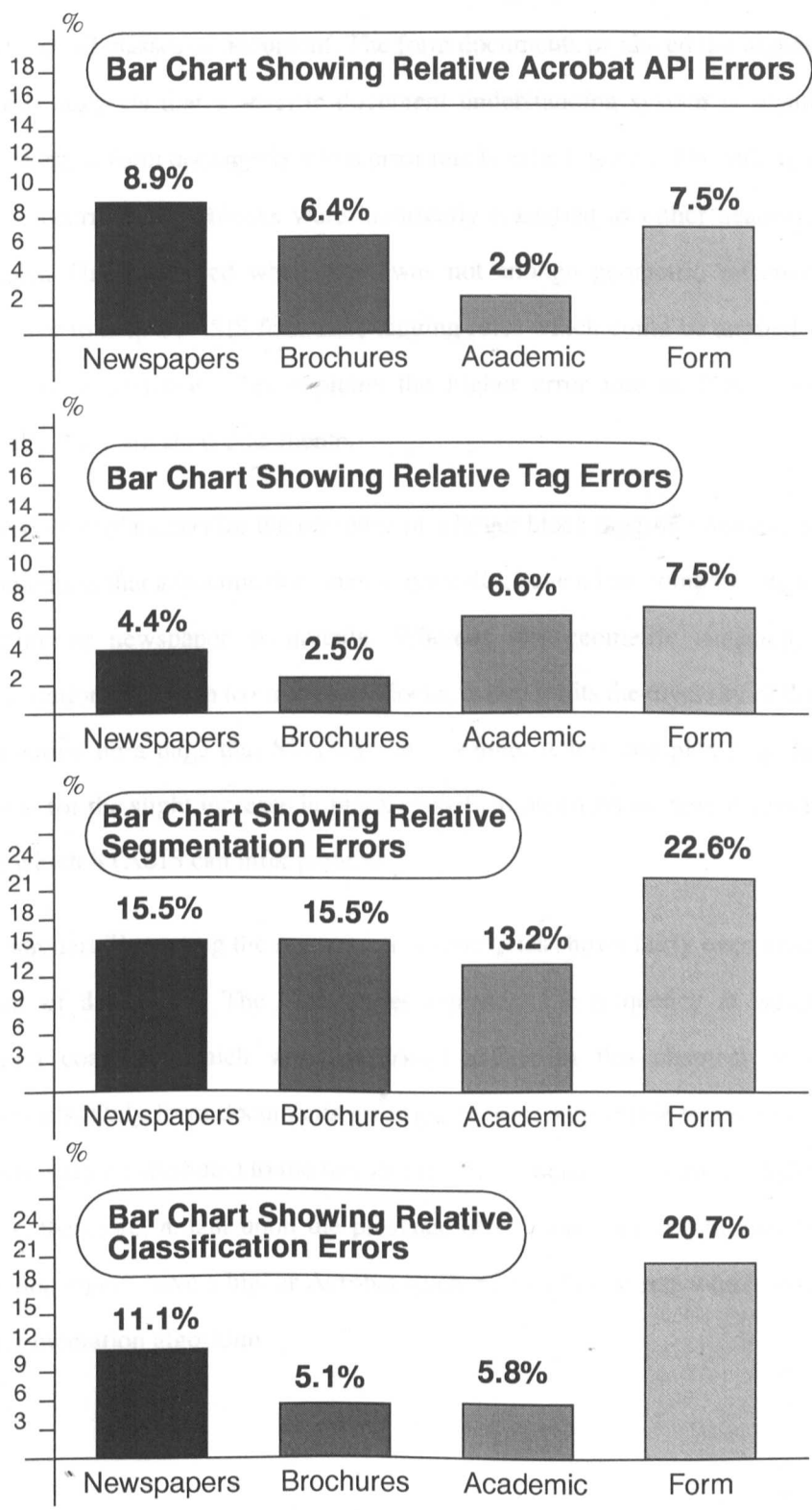


Figure 38: STASIS statistics charts

## Analysis of System Results

---

The bar chart showing block classification (or tagging) errors has produced acceptable results for all classes of document. The form documents produced the highest error rate and this suggests that a specific document understanding system is required for the processing of form documents if this error rate is to be lowered. The bulk of the tagging errors occurred when blocks were incorrectly classified as either headers, footers or captions. This happened when there was not enough geometric information in the document to help STASIS formulate tagging rules which could be applied with a high degree of confidence. This explains the higher error rate in form documents as, typically, they are short documents.

A possible explanation for the presence of a larger block tagging error rate in academic documents is that academic documents, typically, have a less complex page layout than brochure or newspaper documents. Whereas this geometric simplicity helps the segmentation algorithm to create text blocks, it also limits the diversity of the geometric information on a page that STASIS can examine. A less complex page layout could account for the slight increase in block classification errors as there is less information from which STASIS can infer patterns.

The barchart illustrating the segmentation error rates shows fairly even statistics for all groups of documents. The percentages represent the frequency at which the error forming contexts (which were described earlier in this chapter) occur in PDF documents. Only form documents are slightly more vulnerable to segmentation errors and this may be attributed to the fact that the segmentation algorithm is hybrid in origin and is, therefore, reliant upon the presence of columns and text blocks. Furthermore form documents have a higher Acrobat Exchange API error rate which would not help the segmentation algorithm.

## Analysis of System Results

---

The final bar chart displays the most significant results that STASIS produces. The classification performance of STASIS for all academic documents, brochure documents, and newspapers are excellent. The overall classification error rate for the newspaper class is larger than the classification error rate for academic and brochure documents because some newsletters were mis-classified (as either brochures or academic documents).

Unsurprisingly, form documents are the hardest to classify. This is mainly due to the fact that most of the form documents that exist in the world are not pure forms. Frequently they contain instructions at the back of the form which explain how to fill in the form. This has the effect of altering the form's logical and geometric document features to resemble academic and brochure documents. This fact, together with poor API, segmentation and block classification error rates have produced a significantly larger classification error rate for form documents than in any other document class.

### 6.8.2 Evaluating the run-time efficiency of STASIS' strategy

This section presents a breakdown of STASIS' run-times for analysing and classifying a PDF document. Table 22 presents a brief verbal description of a document together with statistics representing the time duration of STASIS' segmentation algorithms (document analysis) and the time duration of STASIS' classification algorithms. For the purposes of this section, the act of classifying a document includes both the advanced analysis level of document process and the classification of the document's feature vector by the neural net. There is little point in differentiating between these two processes as once the neural net has been trained, the act of classifying the feature vector is practically instantaneous.

STASIS is built upon the PDF document model; consequently, the times given cannot be accurately compared and contrasted against other document processing systems as STASIS is reliant (to a certain degree) upon the efficiency of the Acrobat Exchange API.

# Analysis of System Results

The durations listed in Table 22 were calculated from the execution of STASIS upon a Pentium™ personal computer which operates at 133 MHz with 32Mb of RAM, using the Windows95™ operating system. The times are rounded to the nearest second.

| Document Class | Document Description     | Document Analysis Duration (seconds) | Advanced Document Analysis Duration (seconds) | Total Duration (seconds) |
|----------------|--------------------------|--------------------------------------|---|--------------------------|
| Newspaper      | 3 columned, 8 pages long | 15                                   | 4   | 19                       |
|                | 3 columns, 12 pages long | 16                                   | 8   | 24                       |
|                | 3 columns, 6 pages long  | 9                                    | 5   | 14                       |
|                | 6 columns, 52 pages long | 312                                  | 186   | 498                      |
|                | 6 columns, 39 pages      | 213                                  | 163   | 376                      |
| Brochure       | 3 columned, 2 pages long | 2                                    | 1   | 3                        |
|                | 2 columns, 4 pages long  | 5                                    | 2   | 7                        |
|                | 2 columns, 4 pages long  | 2                                    | 2   | 4                        |
|                | 3 columns, 8 page long   | 8                                    | 7   | 15                       |
|                | 2 columns. 3 pages long  | 3                                    | 2   | 5                        |
| Academic       | 1 column, 2 pages long   | 1                                    | 1   | 2                        |
|                | 1 column, 43 pages long  | 19                                   | 13  | 32                       |
|                | 1 column, 34 pages long  | 28                                   | 19  | 47                       |
|                | 1 columns, 18 pages long | 9                                    | 5   | 14                       |
|                | 2 columns, 5 pages long  | 7                                    | 3   | 10                       |
| Form           | 2 pages long             | 3                                    | 2   | 5                        |
|                | 29 pages long            | 28                                   | 19  | 47                       |
|                | 2 pages long             | 2                                    | 1   | 3                        |
|                | 2 pages long             | 6                                    | 2   | 8                        |
|                | 4 pages long             | 5                                    | 3   | 8                        |

Table 22: Run-time performances of STASIS

It can be seen that the advanced document analysis level of processing is significantly faster than the traditional document analysis text-segmentation techniques and that the overall time taken by STASIS to process a document is proportional to the length and complexity of that document’s layout.

## **Analysis of System Results**

---

**Analysis of the errors and error creation contexts has shown that although the decomposition and classification routines can make bad decisions, the data these routines pass to the neural net is often good enough to correctly classify the document. This is very positive indeed. When considering the full cycle of document processing, the document understanding algorithms (which logically follow the classification of a document) have the ability to reconsider (and correct) any decisions made by STASIS which they believe could be erroneous.**

# Chapter 7, Discussion and Conclusions

This chapter draws upon the previous chapters of this thesis to summarise the research that has been made (by the author) into document image processing. STASIS is assessed as a PDF document processing system which attempts to classify documents after building a geometric tree representation of their layout. The practical and theoretical limitations of STASIS are reviewed together with a proposed strategy for universal document image processing in which STASIS' ability to classify documents is vital.

# 7.1 Research synopsis

The research presented in this thesis is unique in the field of document image processing in that it has produced a system which takes a document described by a page description language (and not a bitmap image) as its input. Throughout the development of the prototype and the final system, research has been focused not only on document recognition but also, implicitly, on the suitability of using PDF documents as a document image substitute.

PDF has the ability to describe all the details of a document's layout, for example, the exact position, font and point size of any word in the document. This information is available to any program which can access the Acrobat Exchange API. The immediate advantage that processing a PDF document has, over processing a document image, is that there is a large amount of geometric information instantly available.

It can be argued that PDF is equivalent to a document image which has had its pictures and line diagrams identified as separate entities within the textual content together with the application of OCR and font recognition techniques to the text content portions. This is exactly what Adobe Capture™ performs. It logically follows that this research (particularly the advanced document analysis and document classification sub-systems) is applicable to the field of traditional document image processing, given that a document image is equivalent to a PDF document once a certain degree of pre-processing has been applied to it.

STASIS attempts to identify the logical class of a PDF document, from its layout, in an efficient manner. STASIS is divided into three processing stages: document analysis, advanced document analysis and document classification.

The document analysis technique that STASIS uses to segment the textual content of PDF documents is a hybrid strategy. STASIS takes the logical text lines of a PDF document (which have been formed from a purely data-driven strategy by the Acrobat



## Discussion and Conclusions

---

Exchange API) and forms columns and text blocks from them. The assumption (by STASIS) that there will be columns and blocks present within the document is model-driven, but the model is extremely general and applicable to a wide range of document classes. The union of both data-driven and model-driven strategies makes STASIS' segmentation algorithm robust and reliable but produces greatest success when executed upon documents whose layout consists of text already formatted into columns and blocks.

STASIS uses the output of the document analysis stage as input to a new method of document processing entitled *advanced document analysis*. The overall aim of advanced document analysis is to extract meaningful document features from the document to aid in the classification process. STASIS achieves this by using a blackboard framework which allows knowledge about specific domains of document formatting to be encapsulated within knowledge sources. The knowledge sources are coded with meta-rules which allow them to infer new knowledge (regarding the specific layout of the document which they are processing) each time they analyse a new document. The knowledge sources train themselves by examining example blocks which lie within their domain of knowledge; they infer document specific knowledge rules from the examples and then apply those knowledge rules to the document by tagging blocks. The presence of tagged text blocks helps with the inference of meaningful document features.

The advanced document analysis technique is partly model-driven because it demands the presence of one or more of the following block types: header, footer, super-title, main text body, title, image and caption. In other words, the presence of a document model which contains these types of blocks. Although this strategy is partly model-driven, the model is extremely general and applicable to a large number of document classes.

There are data-driven algorithms present inside some of the knowledge sources, particularly the Column Knowledge Source. The Column Knowledge Source constructs the bounding boxes of columns from the analysis of main text body blocks, tagged as

such by the Text Frequency Knowledge Source. This is a bottom-up strategy: the building of larger entities from the analysis of smaller ones. It would be incorrect to state that the Column Knowledge Source expects columns to be present because the very purpose of its analysis is to test for the presence of columns. As there are two clearly different strategies within the advanced analysis stage of document processing the overall advanced document analysis strategy can be described as being hybrid.

The output of the advanced document analysis is a vector containing the meaningful document features inferred from the PDF document being processed. This vector is the input to a pre-trained neural net classifier which matches the vector to one of four abstract logical document classes: newspaper, brochure, form and academic document. STASIS uses the neural net as a 'black box' classifier; no significant contribution to neural net research has been made by STASIS.

The results produced by the classification sub-system have shown that STASIS can classify brochure, newspaper and academic documents accurately using predominantly hybrid document analysis techniques which extract general document features. This is a significant finding, given that all the document understanding and document classification techniques reviewed in this thesis have been model-driven techniques which seek out and recognise *specific* logical document attributes.

## 7.2 An analysis of STASIS' document processing strategy

One of the most positive aspects of STASIS' advanced document analysis strategy is that it extracts document features rapidly and efficiently whilst performing a level of analysis which contributes to the accumulation of geometrical and logical knowledge about the document. It can be argued that a document understanding system does not need the information generated by STASIS (as a by-product of the document classification process), for example, the elementary tags on the geometric text blocks.

## Discussion and Conclusions

---

However, the author believes that the information provided by the tagged blocks (and the realisation of some of the basic logical relationships between those blocks) can only contribute to the document understanding process. Furthermore, in terms of computational efficiency, advanced document analysis takes only a fraction of the processing time required by the traditional document analysis text-segmentation routines and the neural net (once trained) makes its classification almost instantly.

The advanced document analysis techniques have shown that geometric features (those which describe the layout or format of the document) can be extracted at a basic level of abstraction (for example, the number of pages in a document) or at an advanced level of abstraction (for example, the format of the document's images with respect to the format of the columns in the document) or at any level of abstraction in between these extremes (for example, the number of text styles present in the document). Logical document features, on the other hand, can only be extracted at an extremely basic level of abstraction (for example, the presence of titles in the document). Searching for logical entities of a more detailed nature (for example, a business letter's sender address) would require executing model-driven recognition techniques. STASIS has been deliberately implemented without any purely model-driven recognition techniques so that the results of a non model-driven classification process can be assessed and analysed.

The previous section has highlighted the fact that there is a weak document layout model present within the advanced document analysis strategy. This document model is applicable to all documents which are formatted with images, text columns and text blocks. It follows that STASIS' segmentation and classification techniques produce optimal results when processing documents which correlate strongly with this model, for example, magazines, brochures, newsletters, newspapers, technical documents, journals, books and so on.

The author believed that engineering STASIS to classify documents into a wide range of logical documents (such as those listed in the previous paragraph) would be too difficult to achieve because the level of detail of the logical features extracted from the document was low.

## Discussion and Conclusions

---

Research into the field of document understanding has shown that the only reliable manner of identifying a documents logical class is to seek out and recognise logical entities which are unique to that class of document. Yet many documents have very similar geometric layouts, for example a broadsheet newspaper and a magazine. One could argue that these documents have a similar logical structure and also a similar layout model. It would be very difficult to differentiate between these two document classes using a model-driven approach.

This thesis has shown that a better approach to the classification of these sorts of document would be, firstly, to analyse the document's layout model and secondly to match that layout model to a general logical model. In order to achieve this the author grouped various documents together based on the correlation between their layout and document models, for example, newspapers, magazines and newsletters were grouped into the abstract class of 'newspapers', technical documents, journals, books and manuals were grouped into the class of 'academic documents' and product brochures and pamphlets were grouped into the class of 'brochure' documents. It seems that the documents within these groups exhibit similar traits from both a geometric and logical perspective. A fourth class of document was experimented with: form documents. The aim was to see how a document would be processed which theoretically did not correlate strongly to STASIS' internal document model.

In practice, both the text segmentation strategy and the advanced document analysis strategy were not suited to form documents. Typically and almost paradoxically, the majority of the content in form documents is *not* formatted into text columns and blocks. This did not match the general model of document layout which STASIS was programmed with. As STASIS was not able to either segment form documents efficiently, nor to tag the resulting (badly formed) blocks efficiently, the generated feature vector was ambiguous and not representative of the characteristics of form documents. This seriously impaired the ability of the neural net classifier to make accurate classifications.

## Discussion and Conclusions

---

Furthermore, STASIS does not attempt to generate any useful geometric or logical information from graphic lines. This is a continuation of the hybrid advanced document analysis strategy's model-driven approach; because newspapers, brochures and academic documents conform to the model present in the hybrid strategy, the feature vector created by STASIS is unambiguous. Consequently, there is no need to look elsewhere for further geometric information to help with the classification process. Form documents, on the other hand, are not processed well by the current advanced document analysis strategy, thus their document feature vector is badly formed. Graphic lines are important geometric features of form documents. By ignoring graphic lines, STASIS has ignored a vital source of geometric information in form documents.

Once the target document classes had been chosen, the document feature vector was created. The engineering of this vector was one of the most critical parts of the system development. The feature vector had to be detailed enough to allow the neural net to correlate patterns within the vector to the output classes and yet the vector had to be abstract enough to make STASIS applicable to a wide range of documents. In particular, STASIS had to be applicable to documents which did not correlate 100% to any of the target logical document classes.

The choice of which logical features to put into the feature vector was straightforward; because the target document classes were logical it made sense that all the logical information that could be extracted from the document should be included in the vector. In practice, although the logical features that can be inferred from a geometric document (simply from analysing that documents layout) are extremely basic (essentially just the information about the presence and number of titles in the document), they do exhibit different characteristics for different document classes.

The geometric features (within the document feature vector) represent the geometric document attributes which the author believes accurately portray the layout model of the target logical classes. Certain other features, notably the feature indicating the presence of a document super title, were excluded from the feature vector because the value of

## Discussion and Conclusions

---

these features would be consistent for all the document classes, thus making no contribution to the classification process.

Perhaps the most significant difference between STASIS and other document image processing systems (such as those presented by Niyogi [Niyog94] and Lam [Lam94a]) is that STASIS bases its classification process upon the existence of a correlation between the layout models of certain classes of document and their logical models, whereas Niyogi and Lam's classification processes seek out and identify specific logical attributes. STASIS relies upon the fact that *de facto* formatting rules will be used by the designers of certain documents, for example, newspapers will always be formatted in a multi-column style with many text formats and a high distribution of captioned images.

There is a drawback to this approach; some classes of document will have extremely similar layout models even though their logical models are will be totally different. Consider the following two documents: a newsletter (formatted with two text columns per page, about 15 text styles throughout the document and approximately one image per page) and a detailed product brochure which happens to be formatted in exactly the same fashion as the newsletter. As STASIS relies entirely upon the features extracted from the analysis of a document's layout it stands to reason that two documents which have very similar layouts will have very similar feature vectors. In situations like this, STASIS will fail to make a clear classification decision. Only detailed model-driven logical analysis and semantic content analysis will find any significant differences between these two documents.

## 7.3 A universal document processing system

STASIS has shown that it is possible to logically classify certain sorts of document based on the analysis of their layout model. STASIS has also shown that the class of documents for which this statement is true is limited to documents whose layout model

## Discussion and Conclusions

---

correlates strongly to the layout model which is hard coded into the STASIS system: newspapers, brochures and academic documents. These findings raise two important questions.

Firstly, how can STASIS allocate documents into document classes which are more logically detailed than the current abstract logical target classes? For example, given that STASIS can identify that a document is a member of the base class of newspapers, how can STASIS make a more detailed classification and so state that the document is a magazine?

Secondly, how can the classification technology be integrated with existing model-driven classification to produce a document image processing system which automatically classifies all types of documents, for example, newspapers, letters, forms, invoices, magazines and so on?

The remainder of this section is devoted to answering these questions by providing a brief overview of a universal PDF document processing system which can take any PDF document and reconstruct a logical document by inferring logical information from its layout. The foundation of this universal system is provided by the STASIS blackboard framework and is based upon the results of experiments conducted upon STASIS during the course of this research.

As with all document processing systems, a universally applicable system must segment and analyse the geometric components of the document into a geometric tree. Whereas STASIS executed document analysis as a sub-system, a more appropriate technique would be to integrate the document analysis techniques into a blackboard framework. Knowledge sources should be created which know about characters, words, lines, blocks and a variety of page layout models. The blackboard controller should be programmed with knowledge of a variety of text segmentation techniques (for example, a range of data-driven, model-driven and hybrid strategies), together with knowledge of which technique provides the optimal decomposition strategy in a given context.

## Discussion and Conclusions

---

Furthermore, all aspects of graphic processing could be modelled within this structure. Knowledge sources could be created which know about graphic operators, how graphics are used in page layouts and how graphics are used within the PDF model.

The advanced document analysis stage of processing would remain practically the same as that implemented in the STASIS system. The input to this level of processing would be the geometric blocks and graphics created by the document analysis knowledge sources. There would be only two fundamental improvements to this sub-system.

The first improvement would be the implementation of graphic analysis knowledge sources. The knowledge sources could process the graphics and tag them as either graphics which help define the structure of the page (for example, as inter-column breaks) or graphics that exist as part of a form or as graphics which are a component of a logical diagram.

The second improvement would be the implementation of knowledge sources which can semantically process the content of blocks. In some cases this would involve the natural language processing of textual content, in other cases it would simply be searching for key words or a given text string. The practical drawback of natural language processing is that the system would be restricted to documents written in the English language, although theoretically the system could be implemented with knowledge of as many natural languages as were necessary.

The document vector produced by incorporating the above techniques into the advanced document analysis sub-system would be richer and more diverse thanks to the inclusion of features describing the presence of logical graphics and the presence of more than one kind of subject matter.

So far, the universal document processing system has not differed greatly from STASIS apart from the inclusion of new knowledge about certain geometric elements, the inclusion of new knowledge about the application of decomposition strategies and the restructuring of the document analysis level of processing (in order for it to be integrated



## Discussion and Conclusions

---

into a blackboard framework). However, in order for the universal document processing system to successfully handle all classes of document, the technology which STASIS has developed must be applied together with traditional model-driven logical classification techniques. Once again, the blackboard framework provides a powerful means to achieve this goal, thanks to its ability to model all types of reasoning (or classification) strategies and apply those strategies in an opportunistic manner.

The proposed universal classification system is a two-tier system. Initially, the document feature vector would be generated by the advanced document analysis (ADA) stage of processing. The vector would be passed to a similar classification package to the existing neural net classifier in the STASIS system, in other words a classification system trained with documents from the logically abstract classes of newspapers, brochures and academic documents. In this context a neural net classifier may not be the most suitable classification system to apply as it neither gives a meaningful second or third choice from the classification nor can it reason that the input vector was from an unknown class of documents. It shall be assumed that a more suitable classification system will make the classification and that the document can be classified into one of the following categories: unknown, known-but-the-classification-was-ambiguous or known-and-the-classification-is-unambiguous.

Before any further action is taken, the controller of the blackboard framework should allow model-driven logical classification of the document to take place. Model-driven classification involves the identification of one or more logical components of the document which can support the initial hypothesis that a certain logical document model is present: for example, searching for and finding the sender's address on a letter, plus the sender's signature, would be strong evidence that the document being processed is a letter. However, there should be restrictions on how this is implemented.

Firstly, each document-specific model-driven classification strategy should be implemented within a knowledge source and each of these knowledge sources should have access to a unique knowledge base of rules which can be maintained (by that

## Discussion and Conclusions

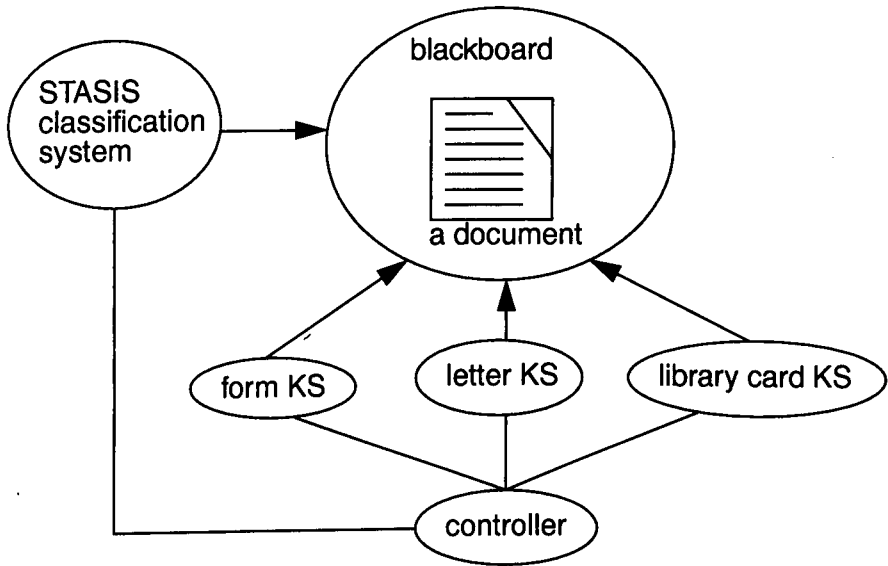
---

knowledge source) in a similar manner to Niyogi's model driven document processing system. Each of these knowledge sources would represent a model-driven strategy for identifying one, and only one, class of document. Each knowledge source should be given the opportunity to analyse the document being processed and to make an assessment as to how much that document correlates with the document model represented by that knowledge source. Furthermore, only document classes which cannot be confidently classified from the feature vector should be handled in this manner, for example, forms, letters and invoices but not newspapers, brochures and academic documents.

Figure 39 provides a representation of this tier of classification within the universal document processing system. Figure 39 is not intended to be comprehensive; instead it is designed to show that the document classes that STASIS cannot classify well can be identified with the help of specialist document-classification knowledge sources. STASIS exists within this system primarily as a classification sub-system. The controller allows each classification system to inspect the blackboard and to execute its own classification algorithms upon the document being processed.

After the activation of all the model-driven classification knowledge sources the blackboard controller must decide which classifier made the best decision. The controller should select that classification process which it believes has found the best match. Since the advanced document analysis (ADA) classification system is in direct competition with model-driven knowledge source (MDKS) classifiers (and vice versa) their evaluations of how close a match they have with the document should be comparable.

As the MDKS classifiers will tend to use highly detailed models within their classification process, no further work is necessary if a MDKS classifier returns the best match. However, the ADA classification technique simply categorises documents into one of three quite abstract document classes: newspapers, brochures and academic documents. From the results of the experiments carried out with the STASIS system, it



*Figure 39: The first tier of the universal document processing system*

has been shown that the classification of documents into highly detailed logical document classes (using the features extracted from the analysis of a document's layout) is unsuccessful thanks to the lack of tangible logical attributes detectable during the analysis process. Consequently, if the ADA classification system recognises that the document being processed is either a newspaper or a brochure or an academic document, some more work must be carried out in order to further classify the document into a more detailed logical class. A second tier of model-driven classification knowledge sources needs to be implemented in order to achieve this. The level of detail of the classification is important; the more detailed the logical class of document is, the better the document understanding of that document will be.

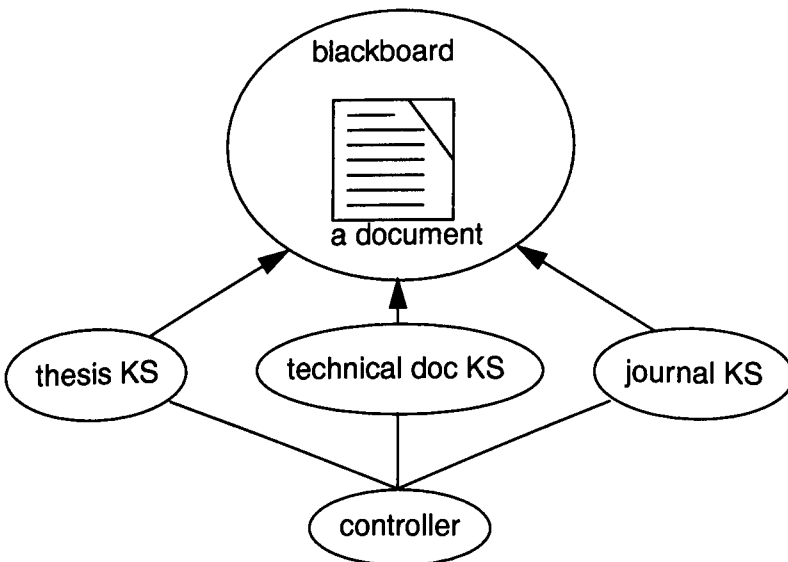
The second tier of the universal document processing system should be implemented in much the same fashion as the first tier; the application of a set of knowledge sources which apply model-driven classification techniques to the document on the blackboard. The knowledge sources should encapsulate a certain document model and base their entire classification process around proving the existence of that document model within

## Discussion and Conclusions

---

the document they are processing. Let us consider the class of academic documents: given that the advanced document analysis classification system has identified the document being processed as an academic document, the controller can now apply the knowledge sources for document models which are *within* the class of academic documents, for example, technical documents, journal documents, fiction books, encyclopedias, conference proceedings and so on.

Figure 40 illustrates the second tier of the universal document classification system. It shall be assumed that STASIS has recognised that the document is a member of the class of *academic* documents. The controller can subsequently invite all the knowledge sources which contain knowledge about *specific* academic document classes (symbolised in Figure 40 by the presence of a thesis KS, a technical document KS and a journal KS) to examine (and classify) the document on the blackboard. In this manner, STASIS helps the universal document processing system to identify efficiently the detailed logical class of the document by narrowing the number of logical classes to which it could belong.



*Figure 40: The second tier of the universal document processing system*

The universal document processing strategy is extensible and scalable. In the same manner in which knowledge sources (which contain knowledge about classifying a particular class of document) can be added to the universal system's blackboard framework, other knowledge sources can be added which contain knowledge about different model-driven processing strategies which are applicable to the same class of document.

The fact that STASIS alone (as a classification system) has weaknesses is not a problem for the universal system. Other classification techniques can be added to the blackboard framework to complement those weaknesses.

## 7.4 Closing remarks and conclusions

STASIS has proven that document image processing techniques are applicable to documents described by page description languages. Furthermore, STASIS has shown that the level of geometric information that page description languages provide is extremely useful for developing detailed document processing algorithms.

Certain aspects of this research cannot be instantly translated to a traditional document image processing system, for example, the analysis and comparison of font metrics. Yet it is the accessibility of detailed geometric information such as this that has allowed STASIS to:

- develop segmentation routines which can efficiently segment a document's text content into geometric blocks;
- develop classification routines which can identify the detailed categories of text blocks within a document without the need for a model-driven approach;
- develop a recognition system which can identify classes of documents which have

## Discussion and Conclusions

---

traditionally been difficult to categorise thanks to the lack of logical and geometric attributes which can verify the presence of a particular document type.

In conclusion, STASIS has shown that

- a hybrid structural analysis system is most suitable for documents which have a loosely defined geometric (or layout) structure. Conversely, documents which have a well defined geometric structure (for example, forms) are optimally processed by a top-down analysis strategy. Consequently, the hybrid strategy employed by STASIS is an effective document processing strategy, yet top-down strategies are faster and more reliable (assuming that the class of document being processed is known);
- the problem of universal document classification is an extremely complex problem requiring knowledge of international formatting conventions (world knowledge), the ability to use the most suitable document processing strategy on a certain type of document (procedural knowledge) and knowledge of how to execute certain document processing strategies (declarative knowledge);
- it is extremely difficult (and perhaps impractical) to classify a real world document into one logical document category. Many documents have a “fuzzy” membership with multiple document classes;
- the strategy developed and utilised (within this thesis) to process PDF documents is commercially viable and that PDF is an excellent medium upon which to base a wide range of new document processing strategies which can utilise the features of PDF which are not instantly available within the medium of traditional bitmap document images.

# References

---

- [Adobe90] Adobe Systems Incorporated. "PostScript language reference manual, second edition" Addison Wesley, 1990
- [Adobe93] Adobe Systems Incorporated. "Portable Document Format reference manual" Addison Wesley, 1993
- [Adobe96] Adobe Systems Incorporated. "Adobe Acrobat FAQ" at <http://www.adobe.com/acrobat/acrofaq.html>, November, 1996
- [Akind93] O. T. Akindele and A. Belaïd. "Page segmentation by segment tracing" in *Proceedings of the Second International Conference on Document Analysis and Recognition*. Japan. October, 1993
- [Akind95] O. T. Akindele and A. Belaïd. "Construction of generic models of document structure using inference of tree grammars" in *Proceedings of the Third International Conference on Document Analysis and Recognition*. Montreal, Canada. 1995
- [Anony96] W3. "Web Style Sheets" at <http://www.w3.org/pub/WWW/Style>, January, 1996
- [Anton95] A. Antonacopoulos and R. T. Ritchings. "Representation and classification of complex-shaped regions using white tiles" in *Proceedings of the Third International Conference on Document Analysis and Recognition*. Montreal, Canada. 1995
- [Baird87] H. S. Baird. "The Skew Angle of Printed Documents" in *SPSE 40th Annual Conference and Symposium on Hybrid Imaging Systems*. Pages 21-47, 1987
- [Barre89] D. W. Barren. "Why Use SGML?" in *Electronic Publishing, Origination, Dissemination and Design*. Vol 2(1) John Wiley & Sons, Chichester. 1989
-

- 
- [Bayer93] T.A. Bayer. "Understanding structured text documents by a model based document analysis system" in *Proceedings of the Second International Conference on Document Analysis and Recognition*. Japan. October, 1993
- [Bayer95] T.A. Bayer and H. Walischewski. "Experiments on extracting structural information from paper documents using syntactic pattern analysis" in *Proceedings of the Third International Conference on Document Analysis and Recognition*. 1995
- [Behle96] B. Behlendorf. "What Is Content Negotiation?" at <http://www.organic.com/Staff/Brian/netscape.html>, January, 1996
- [Belaï93] A. Belaïd and O. T. Akindele. "A labelling approach for mixed document blocks" in *Proceedings of the Third International Conference on Document Analysis and Recognition*. 1993
- [Booch91] G. Booch. "Object Oriented Design with Applications" Benjamin/Cummings, Redwood City, California. 1991
- [Cambe95] I. R. Cambell-Grant. "Standards for document image processing in a multimedia environment" in *Proceedings of Document Image Processing and Multimedia Environments*. IEE, 1995
- [Chene91] Y. Chenevoy and A. Belaïd. "Graphein: hypothesis management for structure document recognition" in *Proceedings of the First International Conference on Document Analysis and Recognition*. Saint-Malo, France. September, 1991
- [Clark95] E. Clarke. "A novel approach to handwritten character recognition" PhD Thesis, University of Nottingham. 1995
- [Commo96] Common Ground Software Inc. "What is Digital Paper" at <http://www.commonground.com>, January, 1996
- [Denge89] A. Dengel and G. Barth. "ANASTASIL: A hybrid knowledge based system for document layout analysis" in *Proceedings of the 11th International Joint Conference on Artificial Intelligence*. Detroit, MI. USA. Pages 1249-1254, IEEE, August 20th-25th, 1989
-



- 
- [Denge93] A. Dengel. "The role of document analysis and understanding in multimedia systems" in *Proceedings of the Second International Conference on Document Analysis and Recognition*. Japan. October, 1993
- [Denge94] A. Dengel. "About logical partitioning of document images" in *Proceedings of the Third Annual Symposium on Document Analysis and Information Retrieval*. Las Vegas, Nevada. Pages 209-218, IEEE, 1994
- [Denge95] A. Dengel and F. Dubiel. "Clustering and classification of document structure - a machine learning approach" in *Proceedings of the Third International Conference on Document Analysis and Recognition*. 1995
- [Dillo93] A. Dillon, C. McKnight and J. Richardson. "Space - The Final Chapter" in *HYPERTEXT, a psychological perspective*. Ellis-Horwood, 1993
- [Drape88] B. A. Draper, R. T. Collins, J. Brolio, A. R. Hanson and E. Riseman. "Issues in the development of a blackboard-based schema system for image understanding" in *Blackboard Systems edited by Englemore and Morgan*. Addison-Wesley, 1988
- [Driva95] D. Drivas and A. Amin. "Page segmentation and classification utilising bottom-up approach" in *Proceedings of the Third International Conference on Document Analysis and recognition*. IEEE, 1995
- [Ellim90] D. G. Elliman and I. T. Lancaster. "A review of segmentation and contextual analysis techniques for text recognition" in *Pattern Recognition*. Vol 23(3/4) Pages 337-346, 1990
- [Engel88] R. S. Englemore, A. J. Morgan and H. P. Nii. "Introduction [to blackboard systems]" in *Blackboard Systems* edited by R. S. Englemore and A. J. Morgan. Addison-Wesley, England. 1988
- [Espos93] F. Esposito, D. Malerba and G. Semeraro. "Automated acquisition of rules for document understanding" in *Proceedings of the Second International Conference on Document Analysis and Recognition*. Japan. October, 1993
- [Espos95] F. Esposito, D. Malerba and G. Semeraro. "A knowledge based approach to layout analysis" in *Proceedings of the Third International Conference on Document Analysis and Recognition*. 1995
-

- 
- [Farro95] G. S. D. Farrow, C. S. Xydeas and J. P. Oakley. "Model matching in intelligent document understanding" in *Proceedings of the Third International Conference on Document Analysis and Recognition*. Montreal, Canada. 1995
- [Goldf90] C. F. Goldfarb. "The SGML Handbook" Clarendon Press, Oxford. 1990
- [Gonza92] R. C. Gonzalez and R. E. Woods. "Digital Image Processing" Addison-Wesley, 1992
- [Graha95] I. S. Graham. "HTML Sourcebook" John Wiley & Sons, Chichester. 1995
- [Ha95] J. Ha, R. M. Haralick and I. T. Phillips. "Document page decomposition by the bounding box projection technique" in *Proceedings of the Third International Conference on Document Analysis and Recognition*. Montreal, Canada. 1995
- [Higas86] J. Higashino, H. Fujisawa, Y. Nakano and M. Ejiri. "A knowledge based segmentation method for document understanding" in *Proceedings of the Eighth International Conference on Pattern Recognition*. 1986
- [Hinds90] S. C. Hinds, J. L. Fisher and D. P. D'Amato. "A document skew detection method using run length encodings and the Hough Transform" in *Proceedings of the Tenth International Conference on Pattern Recognition*. Japan. Pages 464-468, 1990
- [Hiray93] Y. Hirayama. "A block segmentation method for document images with complicated column structures" in *Proceedings of the Second International Conference on Document Analysis and Recognition*. Pages 91-94, October, 1993
- [Hori93] O. Hori and S. Tanigawa. "Raster-to-vector conversion by line fitting based on contours and skeletons" in *Proceedings of the Second International Conference on Document Analysis and Recognition*. Japan. October, 1993
- [Hu93] T. Hu and R. Ingold. "A mixed approach toward an efficient logical structure recognition from document images" in *Electronic Publishing Origination, Dissemination and Design*. Vol 6(4) Pages 457-468, 1993
-

- 
- [Ishit93] Y. Ishitani. "Document skew detection based on local region complexity" in *Proceedings of the Second International Conference on Document Analysis and Recognition* . Japan. Pages 49-52, October, 1993
- [Ishit95] Y. Ishitani. "Model matching based on association graph for form image understanding" in *Proceedings of the Third International Conference on Document Analysis and Recognition*. Montreal, Canada. 1995
- [ISO86] International Standards Organisation. "Information processing - Text and office systems - Standard Generalised Markup Language (SGML)" ISO standard 8879, 1986
- [ISO89] International Standards Organisation. "Information processing - Text and office systems - Office Document Architecture (ODA) and Interchange format" Parts 1-8, ISO Standard 8613, 1989
- [ISO96] International Standards Organisation. "Information technology - Processing languages - Document style semantics and specification language (DSSSL)" ISO Standard 10179, 1996
- [Ittne93] D. J. Ittner and H. S. Baird. "Language Free Layout analysis" in *Proceedings of the Second International Conference on Document Analysis and Recognition* . Japan. October, 1993
- [Iwane93] K. Iwane, M. Yamaoka and O. Iwaki. "A functional classification approach to layout analysis of document images" in *Proceedings of the Second International Conference on Document Analysis and Recognition*. Japan. October, 1993
- [Jacks92] P. Jackson. "An introduction to expert systems (second edition)" Addison-Wesley, 1992
- [Johns83] P. Johnson-Laird. "Mental Models" Cambridge University Press, Cambridge. 1983
- [Keyes94] J. Keyes. "The McGraw-Hill Multimedia Handbook" McGraw-Hill, 1994
-

- 
- [Kise93] K. Kise, N. Yajima, N. Babaguchi and K. Fukunaga. "Incremental Acquisition of Knowledge about layout structures from examples of documents." in *Proceedings of the Second International Conference on Document Analysis and Recognition*. Japan. October, 1993
- [Klir88] G. J. Klir and T. A. Folger. "Fuzzy sets, uncertainty and information" Prentice-Hall International Editions, 1988
- [Klir92] G. Klir. "Probabilistic vs. possibilistic conceptualization of uncertainty" in *Analysis and Management of Uncertainty*, Pages 13-25, Elsevier 1992
- [Lam94a] S.W. Lam. "A computational framework for adaptive reading in document image understanding" PhD Thesis, Center for Excellence in Document Analysis and Recognition, University of New York at Buffalo. October, 1994
- [Lam94b] S.W. Lam. "A local-to-global approach to complex document layout analysis" in *IAPR Workshop on Machine Vision Applications*. Kawasaki, Japan. Pages 431-434, Dec. 13th-15th, 1994
- [Lam95] S.W. Lam. "An adaptive approach to document classification and understanding" in *Document Analysis Systems* edited by Spitz and Dengel. World Scientific, 1995
- [Liu95] Y. H. Liu-Gong, B. Dubuisson and H. N. Pham. "A General analysis system for document's layout structure recognition" in *Proceedings of the Third International Conference on Document Analysis and Recognition*. Montreal, Canada. 1995
- [Loveg95a] W. S. Lovegrove and D. G. Elliman. "Text block recognition from tiff images" in *Proceedings of the Colloquium on Document Image Processing and Multimedia Environments*. Institution of Electrical Engineers, London, 1995
- [Loveg95b] W. S. Lovegrove and D.F. Brailsford. "Document analysis of PDF documents: methods, results and implications " in *Electronic Publishing, Origination, Dissemination and Design*. Vol 8(2 & 3) Pages 207-220, June-September, 1995
-

- 
- [Loveg96] W. S. Lovegrove, D.G. Elliman and D. F. Brailsford. "Advanced document analysis and classification of scanned images" in *The Proceedings of the Fourth European Congress on Intelligent Techniques and Soft Computing*. Aachen, Germany. 1996
- [Nagao88] M. Nagao, T. Matsuyama and H. Mori. "Structural analysis of complex aerial photographs" in *Blackboard Systems* edited by Englemore and Morgan. Addison-Wesley, 1988
- [Nicho92] C. K. Nicholas and L. A. Welsch. "On the interchangeability of SGML and ODA" in *Electronic Publishing Origination, Dissemination and Design*. Vol 5(3) 1992
- [Niyog94] D. Niyogi. "A knowledge based approach to deriving logical structure from document images" PhD Thesis. State University of New York, Buffalo, NY. 1994
- [Niyog95] D. Niyogi and S. N. Srihari. "Knowledge-based derivation of document logical structure" in *Proceedings of the Third International Conference on Document Analysis and Recognition*. Montreal, Canada. 1995
- [Niyog96] D. Niyogi and S. N. Srihari. "Using domain knowledge to derive the logical structure of documents" in *Proceedings of the IS&T/SPIE Symposium on Electronic Imaging*. San Jose, CA. 1996
- [O'Gor93] L. O'Gorman. "The document spectrum for page analysis" in *IEEE Transactions on Pattern Recognition and Machine Intelligence*. Vol 15 1993
- [Oakle88] A. L. Oakley and A. C. Norris. "Page Description Languages: development implementation and standardization" in *Electronic Publishing Origination, Dissemination and Design*. Vol 1(2) 1988
- [Okamo93] M. Okamoto and M. Takahashi. "A hybrid page segmentation method" in *Proceedings of the Second International Conference on Document Analysis and Recognition*. Japan. October, 1993
- [Pavli86] T. Pavlidis. "A vectoriser and feature extracture for document recognition" in *Computer Vision, Graphics and Image processing*. Vol 35 1986
-

- 
- [Pavli91] T. Pavlidis and J. Zhou. "Page segmentation by white streams" in *Proceedings of the First International Conference on Document Analysis and Recognition*. Saint-Malo, France. September, 1991
- [Pavli92] T. Pavlidis and J. Zhou. "Page Segmentation and Classification" in *Graphical models and Image processing*. Vol 54 Pages 484-496, 1992
- [Probe94] S. G. Proberts. "A block-based approach to document formatting and hypertext" PhD thesis, 1994
- [Rice93] S. Rice, J. Kanai and T. Narkter. "An evaluation of OCR accuracy" in *the University of Nevada at Las Vegas Information Science Research Institute Annual Report*. Pages 9-20, 1993
- [Roisi93] C. Roisin and I. Vatton. "Merging logical and physical structures in documents" in *Electronic Publishing Origination, Dissemination and Design*. Vol 6(4) 1993
- [Saito93] T. Saitoh, M. Tachikawa and T. Yamaai. "Document image segmentation and text area ordering" in *Proceedings of the Second International Conference on Document Analysis and Recognition*. Japan. October, 1993
- [Sauvo95] J. Sauvola and M. Pietikainen. "Page segmentation and classification using fast feature extraction and connectivity analysis" in *Proceedings of the Third International Conference on Document Analysis and Recognition*. Montreal, Canada. 1995
- [Shafe76] G. Shafer. "A Mathematical Theory of Evidence" Princeton University, Princeton 1976
- [Sivar95] R. Sivaramakrishnan, I. T. Philips, J. Ha, S. Subramaniam and R. M. Haralick. "Zone classification in a document using the method of feature vector extraction" in *Proceedings of the Third International Conference on Document Analysis and Recognition*. 1995
-

- 
- [Smith93] P. N. Smith, D. F. Brailsford, L. Harrison, S. G. Proberts, D. R. Evans and P. E. Sutton. "Journal publishing with Acrobat: the Cajun project" in *Electronic Publishing Origination, Dissemination and Design*. Vol 6(4) 1993
- [Sperb94] C. M. Sperberg and R. F. Goldstein. "HTML to the max. A manifesto for adding SGML intelligence to the WWW" in *Proceedings of the WWW'94 Conference*. Chicago. 1994
- [Sriha89] S.N. Srihari and V. Govindaraju. "Textual image analysis using the hough transform" in *International Journal of Machine Vision and Applications*. Vol 2(3) Pages 141-153, 1989
- [Sriha92] S.N. Srihari, S.W. Lam, V. Govindaraju, R.K. Srihari and J.J. Hull. "Document Image Understanding" a Center of Excellence for Document Analysis and Recognition Technical Report, University of New York at Buffalo, 1992
- [Sriha94] S.N. Srihari, S.W. Lam, V. Govindaraju, R.K. Srihari and J.J. Hull. "Intelligent data retrieval from raster images of documents" in *Proceedings of the First International Conference on the Theory and Practise of Digital Libraries*. College Station, Texas, USA. June 19th-21st, 1994
- [Statc96] Statcom. "Life On the Bleeding Edge" at <http://www.stratcom.com/edge.html>, January, 1996
- [Strou91] B. Stroustrup. "The C++ programming language, second edition" Addison Wesley, 1991
- [Stutz94] J. Stutz and P. Cheeseman. "A short exposition on bayesian inference and probability" at <http://ic-www.arc.nasa.gov/ic/projects/bayes-group/html/bayes-theorem-long.html>, 1994
- [Sylve95] D. Sylvester, Seth and Sharad. "A Trainable, single pass algorithm for column segmentation" in *Proceedings of the Third International Conference on Document Analysis and Recognition*. Montreal, Canada. 1995
- [Tang91] Y. Y. Tang, C. Y. Suen, C. D. Yan and M. Cheriet. "Document Analysis and Understanding: A Brief Survey" in *The Proceedings of the First International Conference on Document Analysis and Recognition*. St. Malo, France. 1991
-

- 
- [Tang93] Y. Y. Tang and C. Y. Suen. "Document structures: a survey" in *The Proceedings of the Second International Conference on Document Analysis and Recognition*. Japan. 1993
- [Tang95] Y. Y. Tang, H. Ma, M. Xiaogang, D. Liu and C. Y. Suen. "A new approach to document analysis based on modified fractal signatures" in *Proceedings of the Third International Conference on Document Analysis and Recognition*. Montreal, Canada. 1995
- [Taylo93] S. L. Taylor, M. Lípshutz, D. A. Dahl and C. Weir. "An intelligent document understanding system" in *Proceedings of the Second International Conference on Document Analysis and Recognition*. Japan. October, 1993
- [Taylo95] S. L. Taylor, M. Lipshitz and R. W. Nilson. "Classification and functional decomposition of business documents" in *Proceedings of of the Third International Conference on Document Analysis and Recognition*. Montreal, Canada. 1995
- [Toyod82] J. Toyoda, Y. Noguchi and Y. Nishimura. "Study of extracting Japanese newspaper articles" in *Proceedings of the 6th International Conference on Pattern Recognition*. 1982
- [UOM96] University of Maryland . "Document understanding and character recognition WWW server" at <http://documents.cfar.umd.edu>, 1996
- [vanDi80] T. van Dijk. "Macrostructures" Lawrence Erlbaum Associates, Hillsdale: NJ, USA. 1980
- [W3Mag96] W3Magic. "What is W3Magic?" at <http://www.clark.net>, January, 1996
- [WWW96] "Workshop on WWW accessibility for disabled computer users" in *Proceedings of the Fifth International World Wide Web Conference*. Paris. May 6-10, 1996
- [Wahl82] F. M. Wahl, K. Y. Wong and R. G. Casey. "Block segmentation and text extraction in mixed text/image documents" in *Computer Graphics and Image Processing*. Vol 20 Pages 375-390, 1982
-



- 
- [Wasse89] P. D. Wasserman. "Neural computing, theory and practice" Van Nostrand Reinhold, New York. 1989
- [Watan93] T. Watanabe, Q. Luo and N. Sugie. "Structure recognition methods for various types of documents" in *Machine Vision and Applications*. Vol 6 1993
- [Winst92] P. H. Winston. "Artificial Intelligence, Third Edition" Addison-Wesley, 1992
- [Wong82] K. Y. Wong, R. G. Casey and F. M. Wahl. "Document analysis system" in IBM Journal of Research Development. Vol 26 1982
- [Yu93] C. L. Yu, Y. Y. Tang and C. Y. Suen. "Document architecture language (DAL) Approach to document processing" in *Proceedings of the Second International Conference on Document Analysis and Recognition*. Japan. 1993
- [Zadeh83] L. A. Zadeh. "The role of fuzzy logic in the management of uncertainty in expert systems" in *Fuzzy Sets and Systems*. Vol 11 1983
- [Zramd93] A. Zramdini and R. Ingold. "Optical font recognition from projection profiles" in *the Proceedings of the Fifth International Conference on Raster Imaging and Digital Typography*. 1993

# *Appendices*

---

**Appendix I** provides twenty screen shots of the STASIS system processing example documents. Each screen shot is accompanied by explanatory notes.

**Appendix II** is an example DTD and an example of its use.

**Appendix III** is "Text block recognition from tiff images" from the Proceedings of document image processing and multimedia environments, IEE, 1995

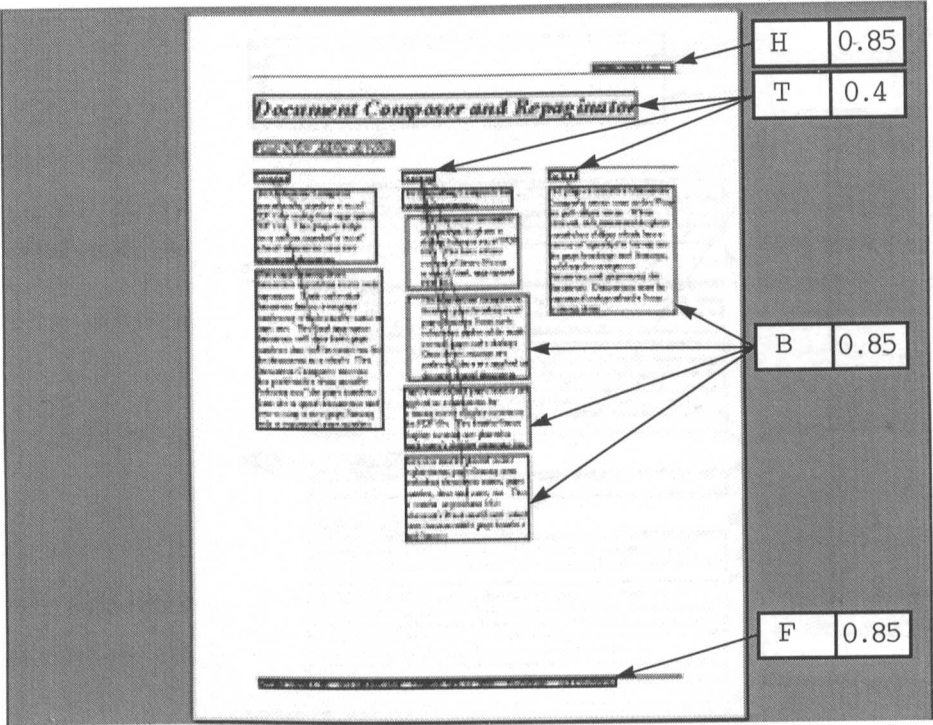
*Appendix I: STASIS*  
*Screen Shots*

---

**Legend**

| Identification tag name | Identification tag, plus tag confidence         |    |     |
|-------------------------|---|----|-----|
| Super Title             | <table><tr><td>ST</td><td>0.7</td></tr></table> | ST | 0.7 |
| ST                      | 0.7   |    |     |
| Title                   | <table><tr><td>T</td><td>0.7</td></tr></table>  | T  | 0.7 |
| T                       | 0.7   |    |     |
| Header                  | <table><tr><td>H</td><td>0.7</td></tr></table>  | H  | 0.7 |
| H                       | 0.7   |    |     |
| Footer                  | <table><tr><td>F</td><td>0.7</td></tr></table>  | F  | 0.7 |
| F                       | 0.7   |    |     |
| Caption                 | <table><tr><td>C</td><td>0.7</td></tr></table>  | C  | 0.7 |
| C                       | 0.7   |    |     |
| Unknown                 | <table><tr><td>UK</td><td>0.7</td></tr></table> | UK | 0.7 |
| UK                      | 0.7   |    |     |
| Body                    | <table><tr><td>B</td><td>0.7</td></tr></table>  | B  | 0.7 |
| B                       | 0.7   |    |     |

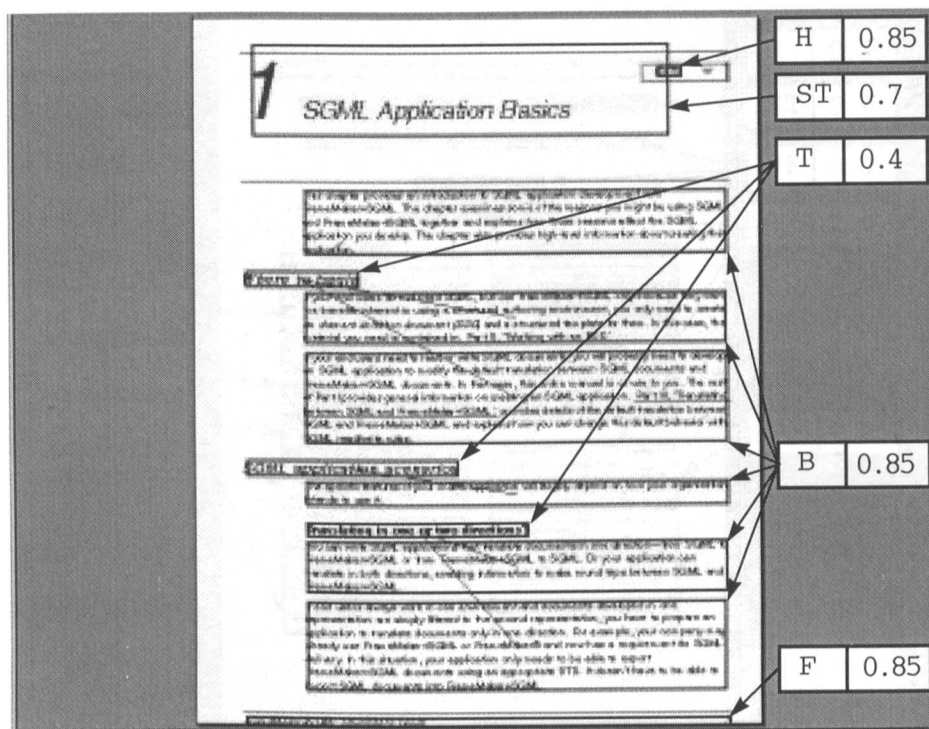
# Screen Shot Title: Academic Example One



**Classification: Academic**  
**Confidence in Classification: 0.9**  
**Notes**

This is an academic document which has a multi-column page layout. There are few text styles, few images, headers and footers are formatted in a consistent manner and there is a consistent column layout throughout the document.

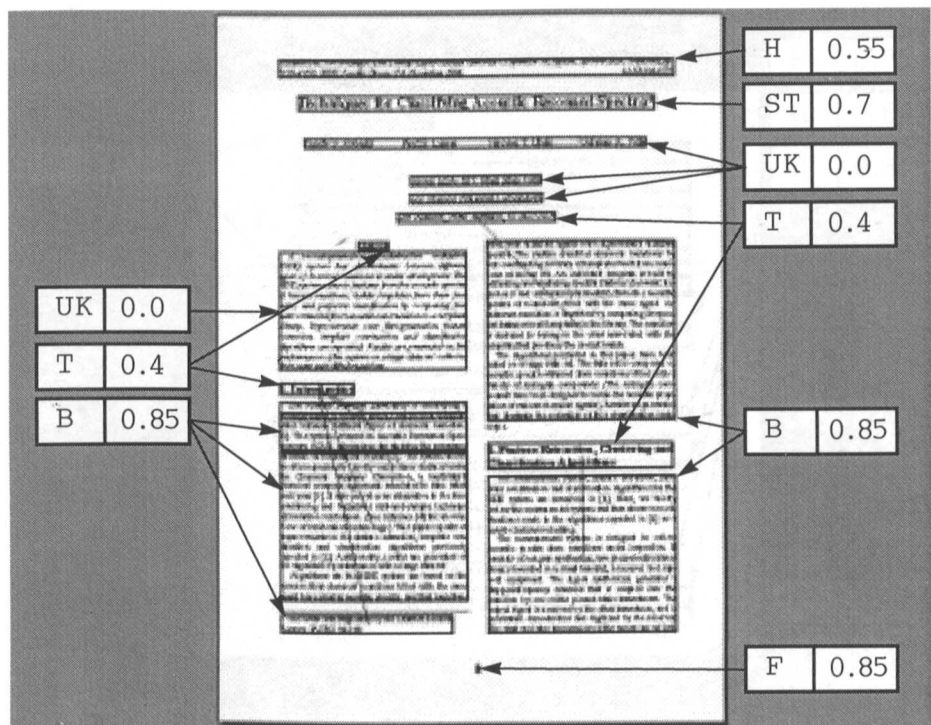
## 23



**Classification: Academic**  
**Confidence in Classification: 0.9**  
**Notes**

This is a technical document with consistent headers and footers and a consistent column style throughout. There are few text and title styles in this document and no images.

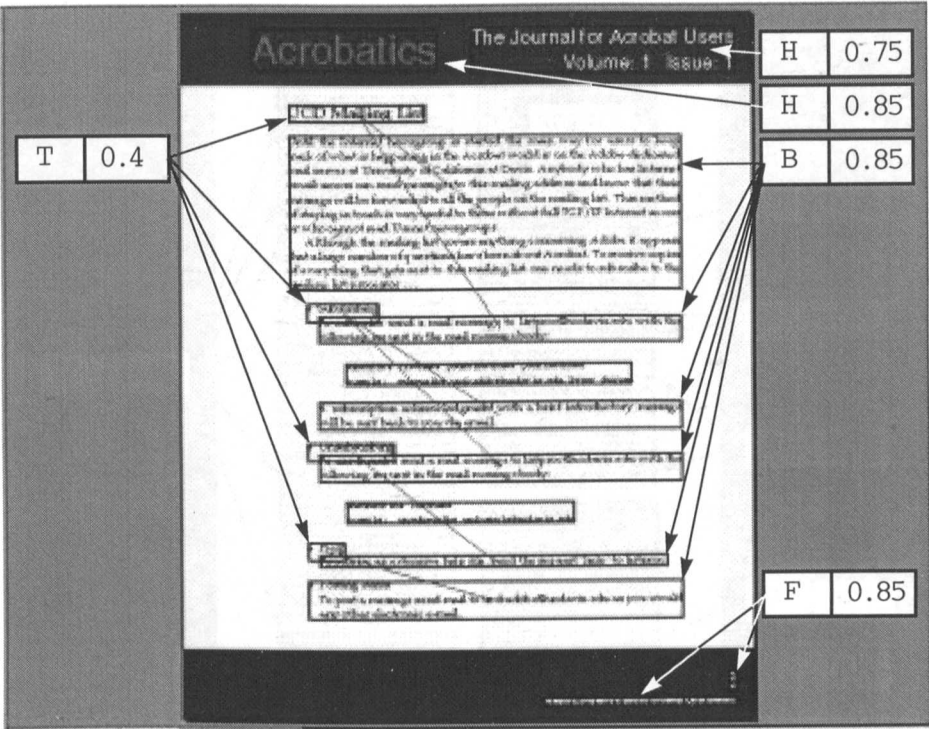
# Screen Shot Title: Academic Example Three



**Classification: Academic**  
**Confidence in Classification: 0.8**  
**Notes**

This is a typical journal document. It is formatted in a multi-column page layout. It contains headers and footers (which are formatted in a consistent style) and few text and title styles. The unknown blocks in this document are high level journal specific entities such as author, abstract and address blocks.

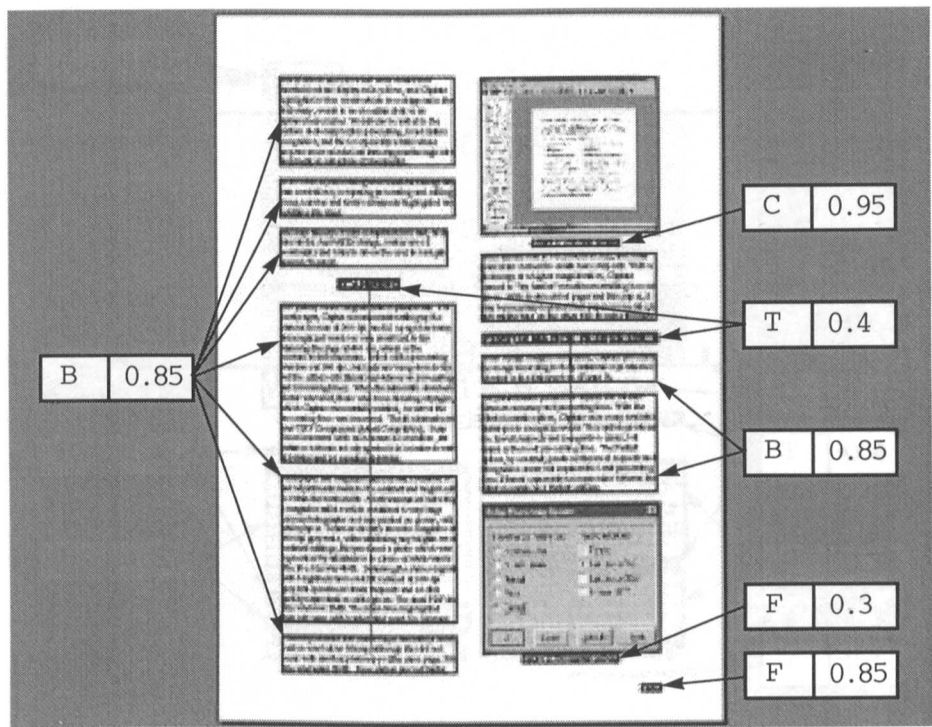
# Screen Shot Title: Academic Example Four



**Classification: Academic**  
**Confidence in Classification: 0.9**  
**Notes**

This is a journal document which displays all the characteristics of an academic document: consistent headers and footers; few text and title styles; few images and a consistent column style.

# Screen Shot Title: Academic Example Five

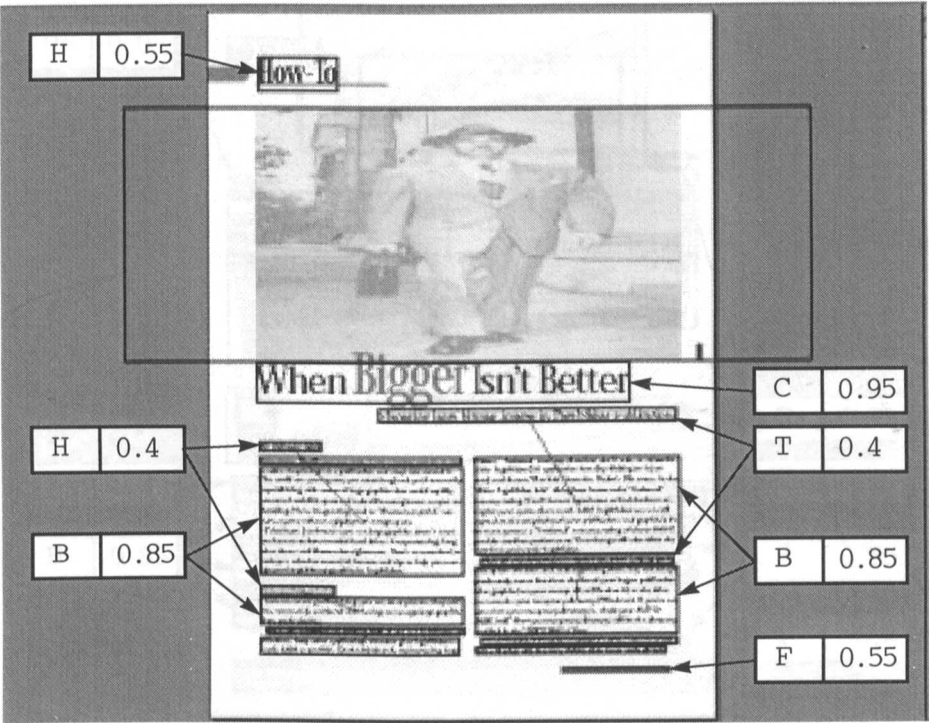


**Classification: Academic**  
**Confidence in Classification: 0.8**  
**Notes**

This is a page of a report which is logically divided into sections. It has few title styles, a consistent column layout and headers and footers formatted in a consistent manner. There are images in this document. STASIS has found captions for these images and these captions are always formatted in the same style with respect to their typeface and the position they occupy relative to their image. Captions which are formatted in a consistent manner are a feature of academic documents.



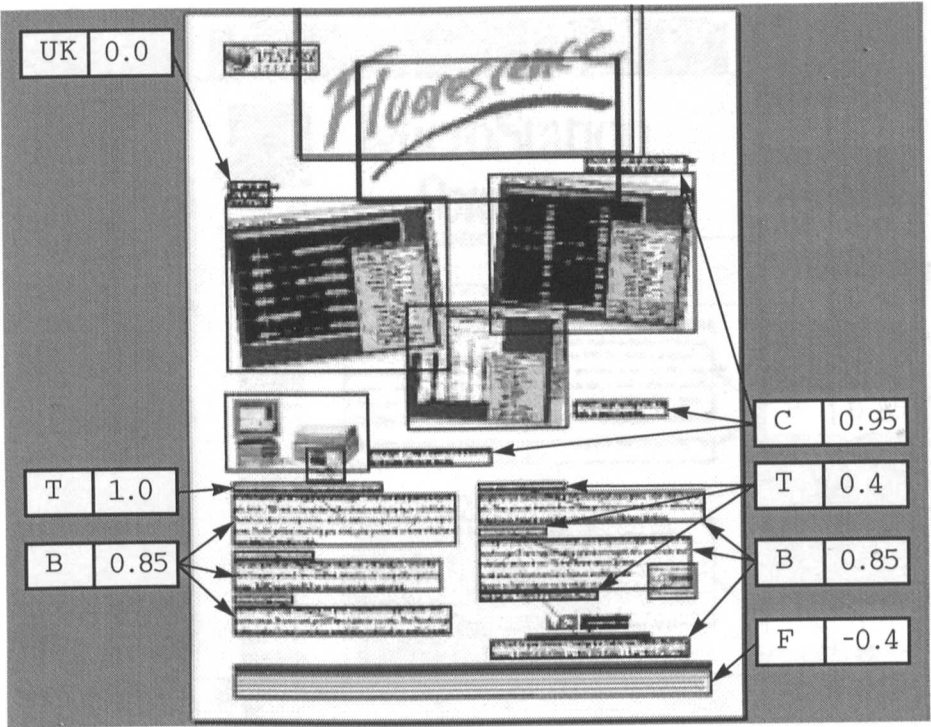
# Screen Shot Title: Brochure Example One



**Classification: Brochure**  
**Confidence in Classification: 0.8**  
**Notes**

This is a brochure which contains an article on bitmap images in documents. Note that the bounding box of the image on this page exceeds the visible boundary of the image. The captions are formatted in an inconsistent style and the document is short, which gives a high average image per page ratio. These are typical features of brochure documents.

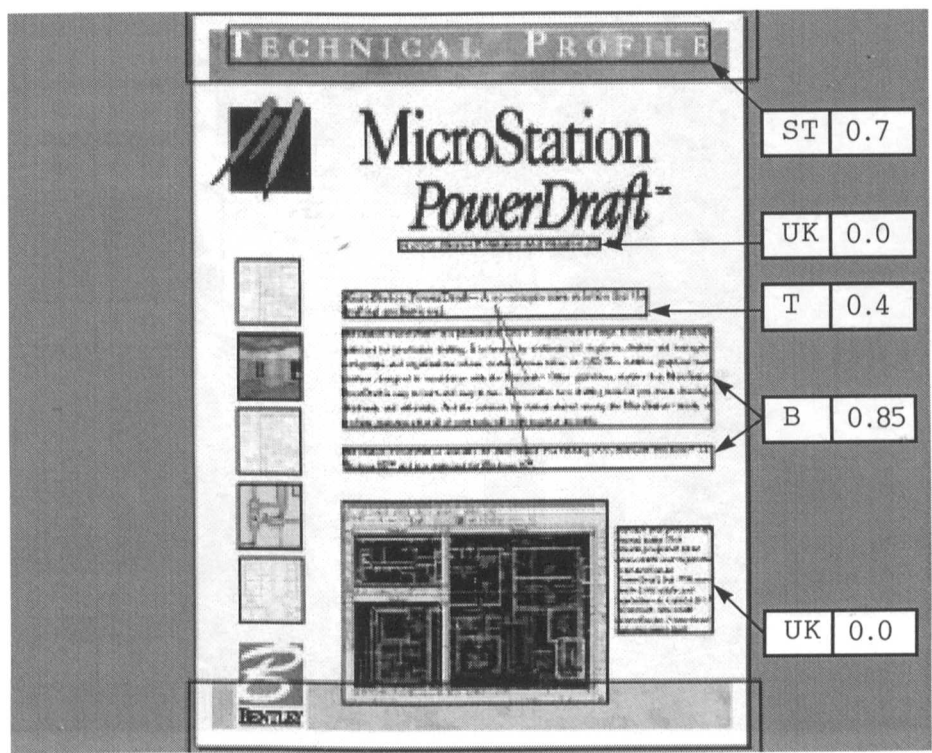
# Screen Shot Title: Brochure Example Two



**Classification: Brochure**  
**Confidence in Classification: 0.8**  
**Notes**

This is a classic example of a brochure which advertises a product. It is only one page in length, with many images. The frequency of images per page could be the most significant feature of a brochure. Two blocks in this image deserve special attention. There is a footer block which has a negative confidence. This is brought about by the lack of footer examples in the document (because the document is one page long) which STASIS can use to create rules for inferring footers. Thus STASIS is not confident that it is a label. The other block worth mentioning is a title block on the left of the image which has a tag confidence of 1.0. This has arisen because the title knowledge source was asked to verify its tag assertion when the controller was resolving a conflict between the title KS and the caption KS.

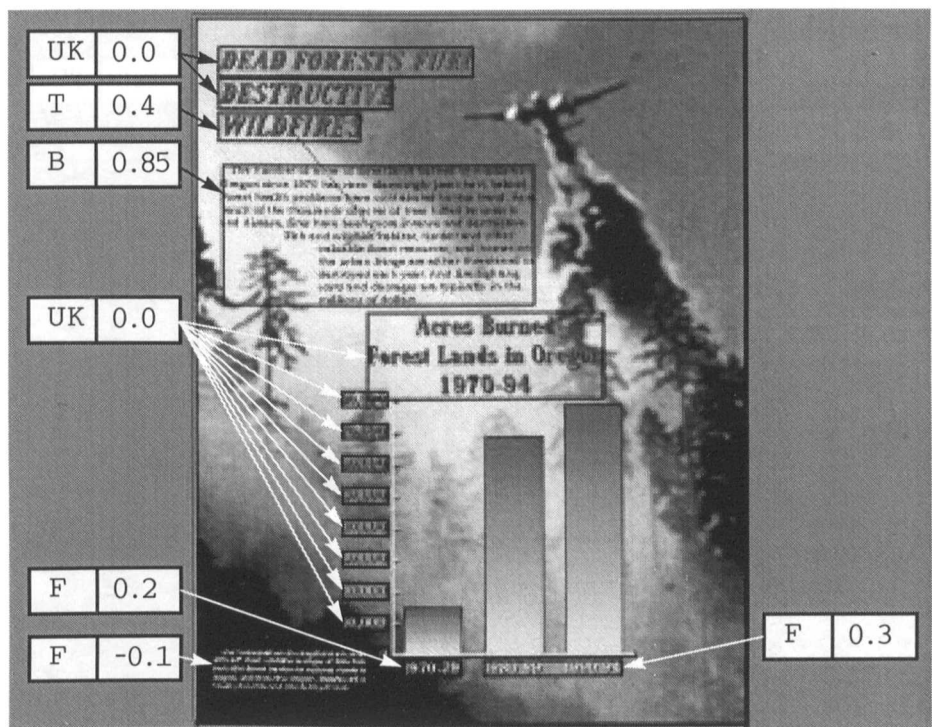
# Screen Shot Title: Brochure Example Three



**Classification: Brochure**  
**Confidence in Classification: 0.8**  
**Notes**

This is a brochure document which has many images and an inconsistent column style. There are relatively few text styles and few pages. Notably the major titles of the document have not been processed by STASIS. They are in-line PDF images. In-line images occur infrequently and typically contain textual information - as in this case. Processing these 'word' bitmaps as images would be bad, as STASIS would try and find captions for them and the number of block mis-classifications would increase.

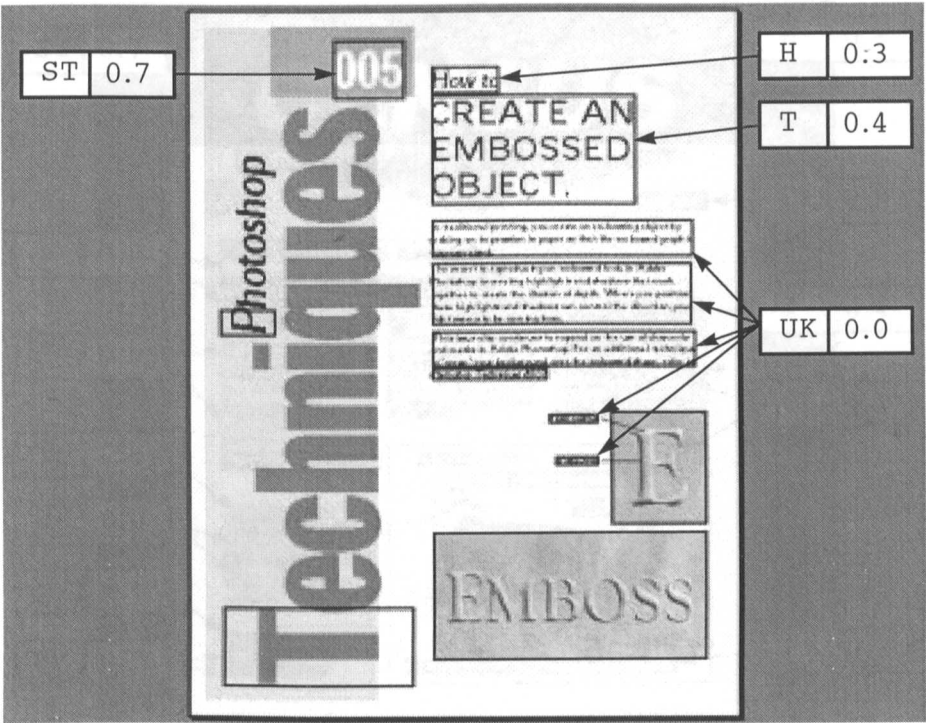
# Screen Shot Title: Brochure Example Four



**Classification: Brochure**  
**Confidence in Classification: 0.6**  
**Notes**

This brochure contains a graphic bar chart which consists of text items and graphic items. The text in the bar chart has been tagged as either unknown or mis-tagged as footers, although the confidence of the mis-classifications is low. This example illustrates STASIS' inability to process graphics which form abstract logical entities. There are three blocks at the top left hand corner of this example which ideally should have been segmented into one block, but the document author incrementally increased the point size of each line thus making them unique text styles. Only one of the three blocks has been correctly classified as a title.

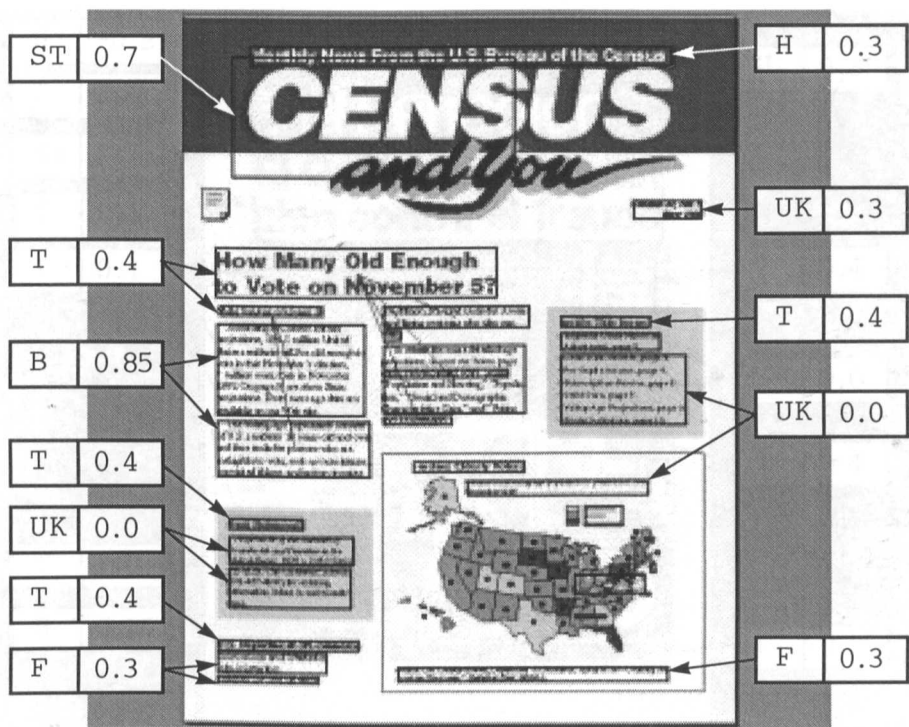
# Screen Shot Title: Brochure Example Five



**Classification: Brochure**  
**Confidence in Classification: 0.8**  
**Notes**

This is a typical brochure document. There are many images throughout the document. There are blocks tagged as unknown on this page because there is another text style present in the remaining pages of the document which has been declared as the main text style by the Text Frequency Knowledge Source. The super title is '005' because of this block's dominant typeface properties.

# Screen Shot Title: Newspaper Example One

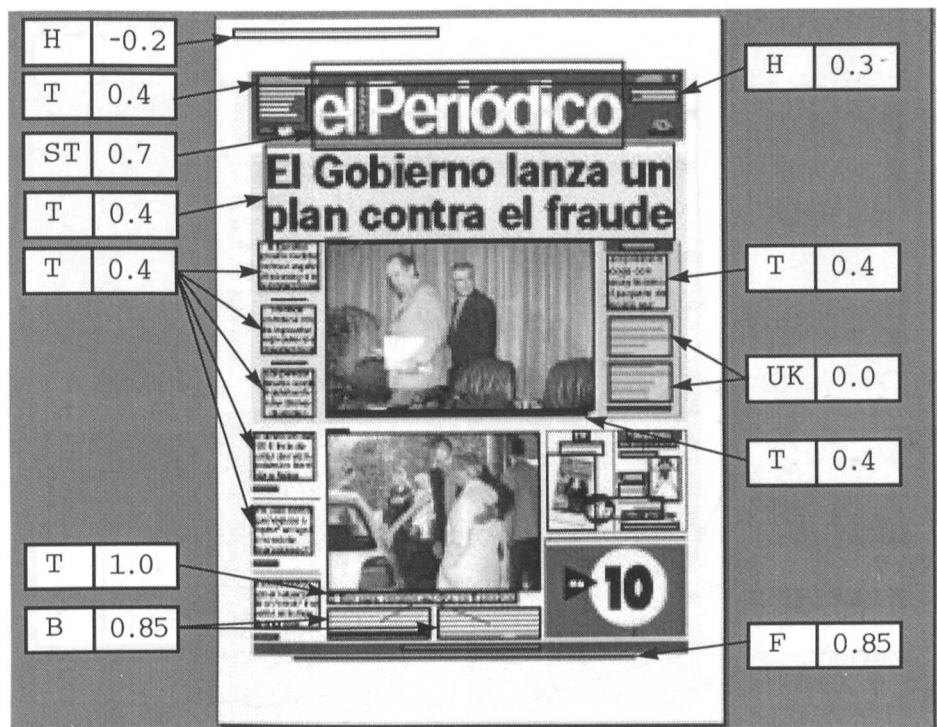


**Classification: Newspaper**  
**Confidence in Classification: 0.7**  
**Notes**

This document is not a typical broadsheet newspaper. It is a newsletter with the same logical structure as a newspaper: a document consisting of many articles. This is the front page which contains the super title (although the bounding box of the title is only the first quad of the word) and article, a graphic diagram and 'leads' to other articles later in the document. The diagram and the leads are beyond the recognition capacity of STASIS. They are either classified as unknown or in the case of the diagrams caption mis-classified as a footer. These mis-classifications have not affected the overall classification of the document.



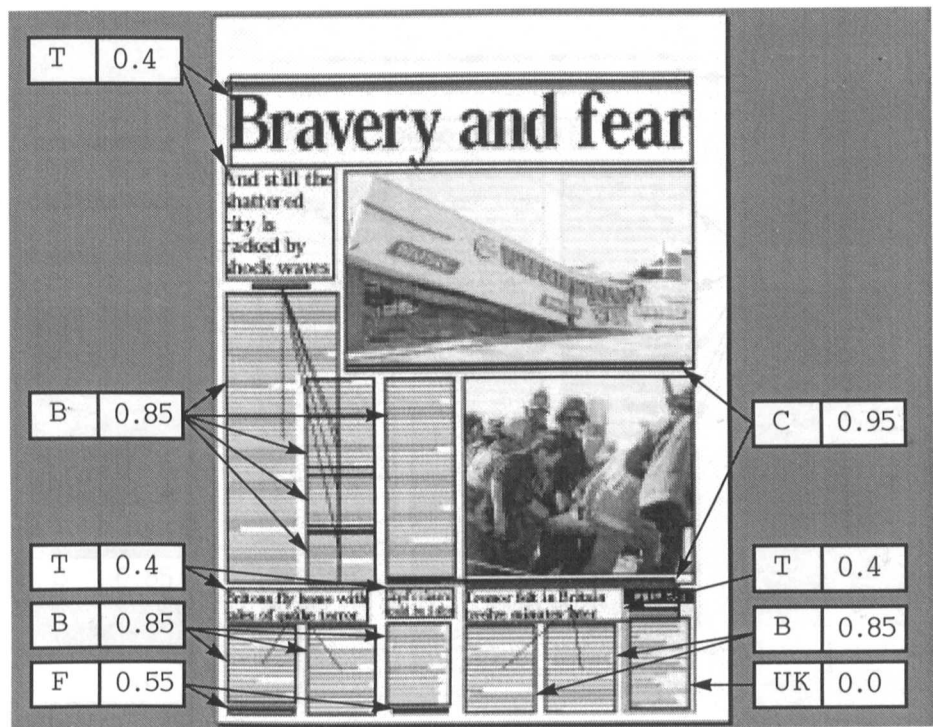
# Screen Shot Title: Newspaper Example Two



**Classification: Newspaper**  
**Confidence in Classification: 0.9**  
**Notes**

This is a Spanish broadsheet newspaper which has been segmented and classified by STASIS, thus proving STASIS’ independence from language and semantic content. This page shows ‘leads’ to other articles which have been identified as titles. There is a mis-classification present. The main image caption has been mis-classified as a title block. This is due to the lack of a standard caption patterns throughout the document from which STASIS can infer formatting rules. The inconsistency of captions, columns styles and large numbers of text and title styles are all features of newspapers.

# Screen Shot Title: Newspaper Example Three

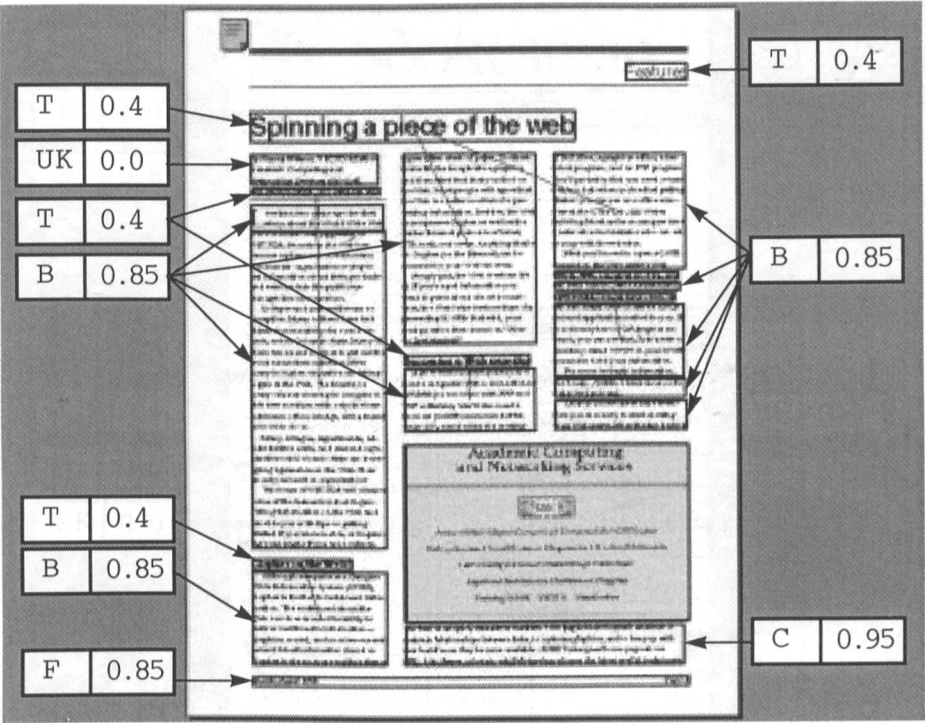


**Classification: Newspaper**  
**Confidence in Classification: 0.9**  
**Notes**

This is a page from the Evening Standard which illustrates STASIS' ability to segment multi-column pages when given good text lines from the Acrobat API to work with. The image captions in this document have been correctly classified.



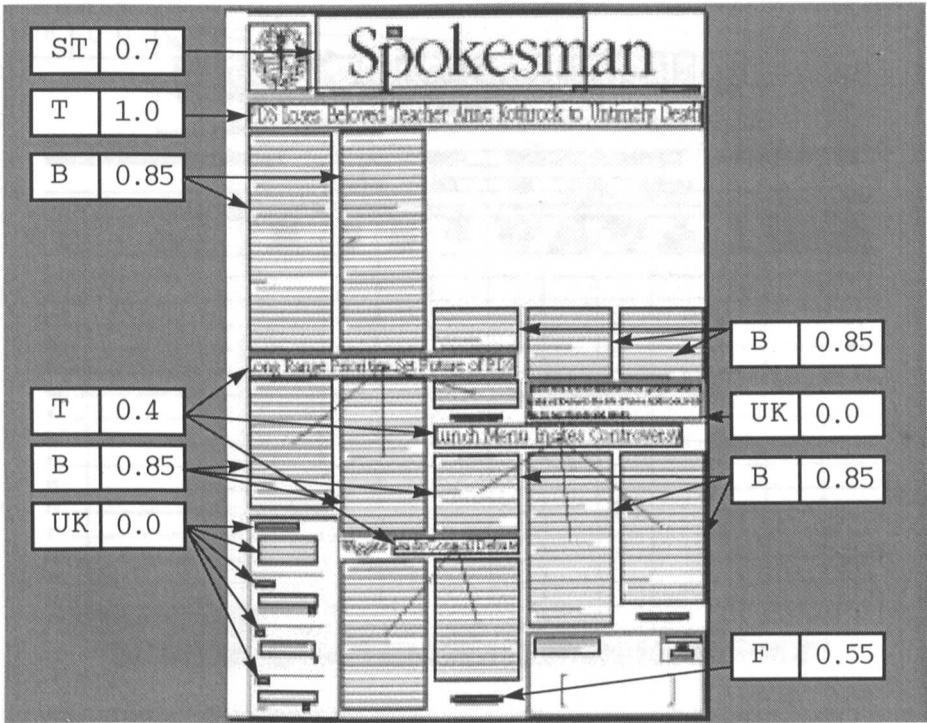
# Screen Shot Title: Newspaper Example Four



**Classification: Newspaper**  
**Confidence in Classification: 0.6**  
**Notes**

This is a newsletter. It has slightly more geometric structure in its layout than a newspaper or a magazine. It has consistent column layouts, consistent headers and footers, fewer text and title styles and fewer images than a newspaper. However, there are still enough differences in the magnitude of these features from other documents to give a correct document classification, albeit with a lesser confidence.

# Screen Shot Title: Newspaper Example Five



**Classification: Newspaper**  
**Confidence in Classification: 0.7**  
**Notes**

This example document has been prepared for the internet by removing all the document's images in order to keep the file size down. The absence of images has not affected the document classification, just the degree of confidence of the classification. The column layout and the text and title styles are enough to accurately classify this document. STASIS has been trained with examples of newspapers and newsletters which do not have images in them.

### Screen Shot Title: Form Example One

The screenshot shows a software interface for tracking video games. At the top left, there is a box containing the text "UK" and "0.0". Below this, a menu bar contains the following options: "Report of Bar-Coded Games Played", "Destroyed", and "Lost". Below the menu bar, there is a large grid table. The grid has several columns, with the first column containing numbers 1 through 10. The other columns contain text labels, which are mostly illegible due to the low resolution. The grid is mostly empty, with some faint text visible in the header row.

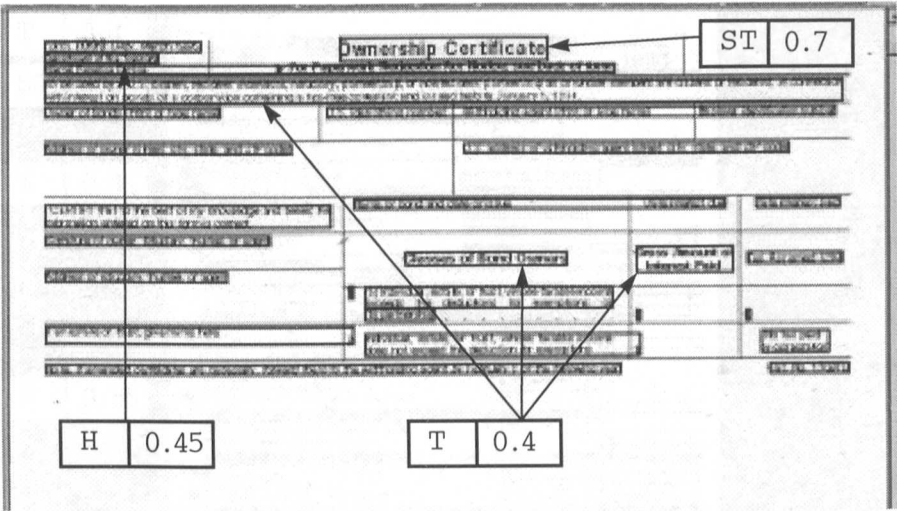
**Classification: Form**

**Confidence in Classification: 0.8**

## Notes

This document has been classified as a Form document simply on the strength of the feature which describes the number of graphic lines on the page. Relying on a single feature is not a good manner of identifying an entire class of document. The segmentation process has been confused as this document was created in one orientation and then printed (to PDF) in a second orientation. The bounding boxes of the words have all been rotated through ninety degrees. Thus the top of the bounding box is the now the left, the left is the bottom, the bottom is the right and the right is the top. STASIS has not been programmed to handle this anomaly.

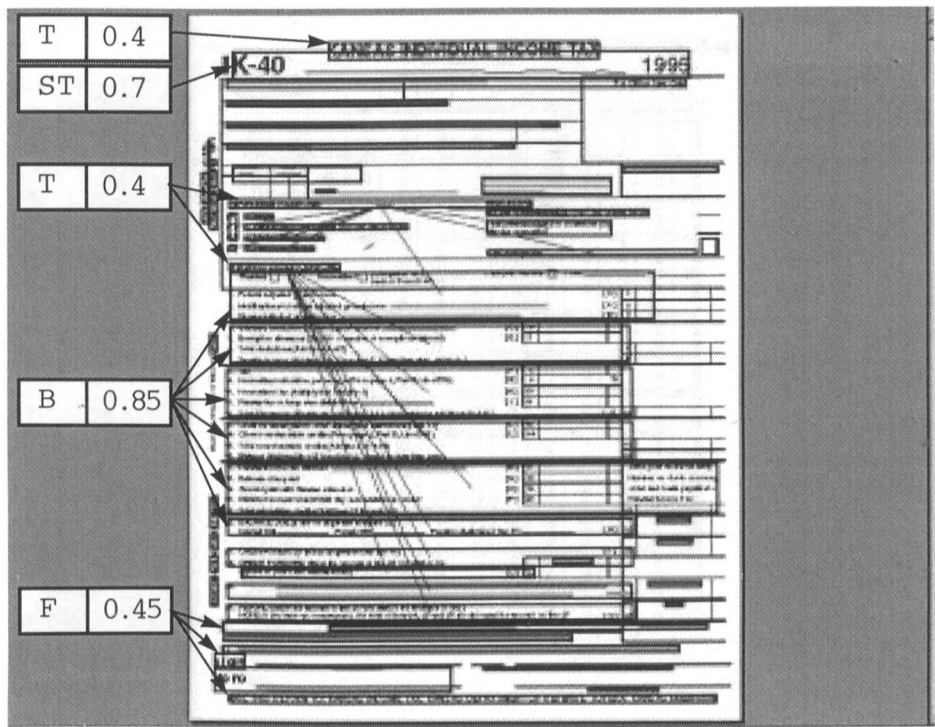
# Screen Shot Title: Form Example Two



**Classification: Form**  
**Confidence in Classification: 0.1**  
**Notes**

STASIS has been programmed to look for document layout features primarily in brochure, academic and newspaper type documents. This document image shows poor block segmentation (the text lines cross vertical graphic lines in places) and only a few blocks are not classified as unknown. There are other features which are typical of forms: no images, few text and title styles and an irregular column style. However, these are features of many academic documents as well.

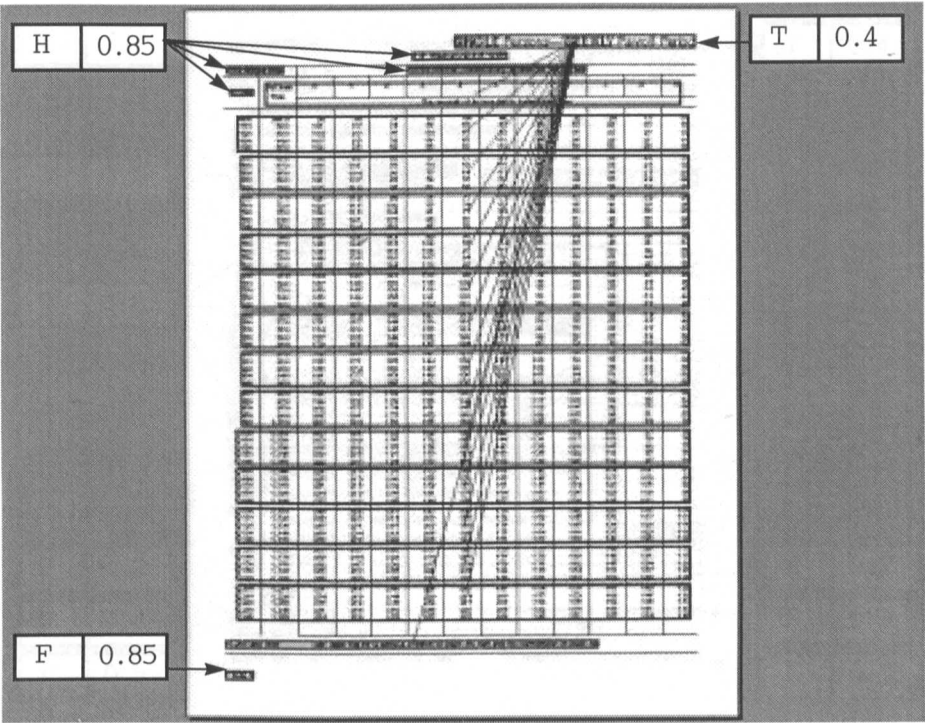
# Screen Shot Title: Form Example Three



**Classification: Form**  
**Confidence in Classification: 0.9**  
**Notes**

This document is a typical form document found on the internet. Of particular interest are the vertical text lines found on the left of the document. STASIS does not segment or process vertical text well.

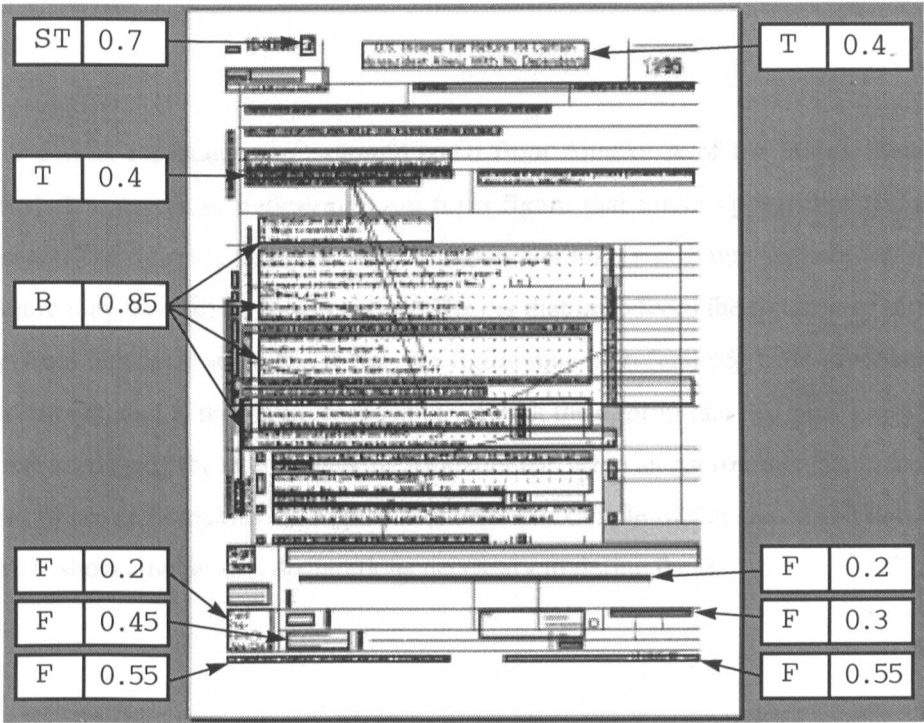
# Screen Shot Title: Form Example Four



**Classification: Academic**  
**Confidence in Classification: 0.3**  
**Notes**

This document was classified (correctly) as an academic document. The image above shows a single page from this document. STASIS could, with minor alterations, detect the presence of a form on this page and record this information. This would be valuable information to pass on to a document understanding system, for example, “this document is academic, but page X is a form”.

# Screen Shot Title: Form Example Five



**Classification: Form**  
**Confidence in Classification: 0.9**  
**Notes**

This form is blank; it is not ‘filled in’. Consequently, all the inferences that STASIS makes regarding the class of the blocks (for example, if it is a title or not) is made from data collected from analysing an empty form. Even though many of these classifications are correct, they are made from incorrect inferences.

## Appendix II: An example DTD

---

This appendix reproduces an example taken from Annexe A of the SGML standard [ISO86]. The aim is to define mark-up for a figure that might appear in a technical document. The figure body may consist of artwork or text (which may include lists), and the figure may have an optional caption. The tag that introduces the figure may include an optional identifier attribute, so that the figure may be referenced from elsewhere in the document, and if the figure consists of artwork the appropriate tag must include an attribute specifying the size, so that the formatter can leave an appropriate gap in the text for the figure to be pasted in. Figure A shows an example of the use of the tags, and Figure B shows the SGML declarations necessary to define them.

```
<fig id=babel>
<figbody>
<artwork depth=3in>
<figcaption>The Tower of Babel by Pieter Brueghel (1563)
</fig>
```

*Figure A: An SGML mark-up for a figure*

```
<!ELEMENT fig -- (figbody, figcap?)>
<!ELEMENT figbody -O (artwork | (p | ol | ul)+)>
<!ELEMENT artwork -O EMPTY> <!ELEMENT figcap -O (#PCDATA)>
<!ATTLIST fig id ID #IMPLIED>
<!ATTLIST artwork depth CDATA #REQUIRED>
```

*Figure B: SGML DTD declarations for a figure mark-up*

In Figure B we see first declarations of the elements to be tagged in a figure. Line 1 declares an element `fig` (and hence the associated tag `<fig>`) and asserts that it consists of an obligatory `figbody` and an optional `figcap`. The two dashes indicate that both open and close tags are required for a `fig`. The next line defines a `figbody` as either artwork or

---



---

an indefinite number (at least one) of occurrences of a paragraph, ordered list or unordered list. (It is assumed that there are already definitions for paragraph (<p>), ordered list (<ol>) and unordered list (<ul>)). The characters -O signify that the start-tag is required, but that the end-tag can be omitted so long as its presence can be unambiguously inferred from the context. Line 3 specifies that artwork has no content (it is something that will be provided outside SGML), and line 4 defines figcap as an arbitrary string of characters. (PCDATA indicates that the string will be parsed by the SGML parser, and therefore may include entity references.) Finally we have two declarations that specify attributes to tags. The first of these says that fig has an optional attribute with name id, of type ID a code denoting a unique identifier. The second says that artwork has an obligatory attribute named depth, whose value is a character string. CDATA indicates that this string will not be processed by the SGML parser.

## *Appendix III: RECOG*

---

# TEXT BLOCK RECOGNITION FROM TIFF IMAGES

William Lovegrove, David Elliman

## Abstract

*The reproduction of a scanned document should include not only the optical character recognition of text, but also the structure of that text on the page and the appearance of that text itself (i.e. font recognition). This is a paper that presents an algorithm which structurally recognises the text of a page image. The method is based upon the "Docstrum plot" algorithm by L. O'Gorman[1]. Modifications have been made to O'Gorman's algorithm which render very good results at identifying paragraphs and lines in particular. The algorithm implementation can, to a limited degree, describe the logical relationship of the text elements of the original page. The limitations of the algorithm are due to the lack of information available without O.C.R. and font technology incorporated into the algorithm implementation. The algorithm implementation has a graphical interface which portrays the state of the algorithm during the process of decomposition.*

**KEYWORDS** DOCUMENT UNDERSTANDING, PAGE DECOMPOSITION, BLOCK RECOGNITION, TEXT RECOGNITION.

## 1 Introduction

With the emergence of structured documents over the internet (HTML) and the subsequent increased usage of logical hypertext links and logical text searches within document databases, there is an increasing need to recognise the logical structure of a document as well as the semantic content of a document during the process of document analysis. The two areas of document understanding are not reliant upon one another and have so far existed without needing one another. To date the majority of research into document understanding has been conducted in the field of character recognition with only a small emphasis dedicated to looking at the logical models that exist today.

This paper presents an algorithm which decomposes a page into its component blocks based upon a nearest-neighbour clustering algorithm. The blocks which this clustering algorithm produces are classified into text and non-text types. The text type blocks are then processed by an adaption of the clustering algorithm presented by O'Gorman which is tailored to finding logical paragraphs, lines and words in the text blocks.

The logical structure of a document is required prior to further complex processing in documents which do not have a linear structural order: newspapers and magazines.

## 2 Page decomposition

### 2.1 Preprocessing by vectorisation

The images processed by this system are TIFFs<sup>1</sup>. A vectorisation technique developed at the University of Nottingham by Prof. David Elliman reduces the image to a series of vector loops. One vector represents the boundaries between pixels of different colour. These loops are placed within a two dimensional linked list data structure which implicitly stores the positional information of the loops within its structure. The letter 'g' consists of three loops. At the top level the outer perimeter of the 'g' is represented by a single loop of vectors. Before continuing on to look at the next character or blob on the page the vectorisation algorithm looks inside the perimeter of the loop it has just found for other pixel value differences. In the case of a 'g' there are two more loops inside the outer perimeter, which are placed in the second dimension of the data structure – accessible only through the vector loop representing the outer perimeter of 'g'. All details known about the vector loops are stored with the loops: lower left position, height and width.

By abstracting the results of the vectorisation programme the algorithm can place a point on the Cartesian co-ordinates of the page for every loop it finds in the primary dimension of the data structure. The inner loops of the data structure are not processed, reducing the amount of redundant information to be processed.

In keeping with common O.C.R. page preprocessing practice the page is checked for skew and corrected prior to further processing by the algorithm. Baird[2] suggests a skew detection algorithm which involves abstracting each character to reference point from which energy points are calculated through a range of conceivable angles. This technique seemed the most applicable after the vectorisation had already reduced the character on the page to abstracted loops.

### 2.2 Block isolation

Page vectorisation produces a set of abstracted points in a Cartesian space. It would be highly advantageous to identify regions of these abstracted points as regions of similar content. For example a region consisting of text only. This process is known as auto zoning. Block classification would allow the algorithm to execute later stages of the processing efficiently. No time would be wasted attempting to apply text recognition techniques to half-tone images. By using O'Gorman's algorithm loops can be clustered together based upon their physical location within the Cartesian space in relation to their neighbours. Although O'Gorman's algorithm searches for five neighbours in its clustering algorithm, it was found that this amount of searching for neighbours was both computationally expensive and the information provided by the fifth and fourth neighbours was never used.

The algorithm presented here looks for the three closest neighbours in any direction over any distance. This was found to cluster together loops of a similar classification. The algorithm is based upon the hypothesis that the three nearest neighbours in a vectorised document are highly likely to be of the same "type" as the seed loop. Furthermore, there is a high probability that the three nearest neighbours are physically within the same logical area of the page. This hypothesis is not as sound when four or five neighbours are considered.

At this stage of the algorithm the "k-clustering" technique works for both text and graphics. The vectorisation process typically leaves images as clusters of "blob" loops. The vec-

---

<sup>1</sup>Tagged image file format

torisation technique thresholds the TIFF image to obtain a monotone image so that it can easily identify the pixel boundaries.

## 2.3 Block classification

The blocks are classified by analysing the horizontal projections of the contents of the blocks. Lines of text have a characteristic ‘fingerprint’ projection. This makes it easy to identify blocks of text from other block types. The ‘fingerprint’ for images is not sufficiently unique to differentiate image blocks from diagram blocks. Generally speaking the ‘fingerprint’ of line diagrams are too varied to classify accurately from image ‘fingerprints’.

Many other block classification techniques exist all of which look to extract features from blocks to use in statistical classification methods[3].

## 3 Tailored clustering for text blocks

The algorithm encompasses several passes in which subtle changes are made to the clustering technique according to the logical level being extracted from the set of vector loops. Simply by repeating the clustering algorithm presented by O’Gorman with a value of  $k=1$  or  $k=2$  was not enough to isolate lines of text. Knowledge in the form of restrictions upon the searching for neighbours was imparted to the algorithm. In order to find lines, O’Gorman’s algorithm was adjusted in the following manner for each block identified as comprising of text.

1. X-sort all the vector loops within the block in X increasing order. Thus, once a loop that satisfies the remaining criteria is found, it is possible to stop searching the list for other candidate neighbours as it is guaranteed that the closest loop has been found.
2. Remove the first vector loop in the sorted list. This becomes the first ‘seed’<sup>2</sup> character of the first line.
3. Search along the list (in X increasing order) for characters that are inside a search angle of 15 degrees above and below the horizontal projection of the centre of the previous ‘seed’ character. The character being tested must *also* be within a certain vertical distance of the previous ‘seed’ character. The vertical distance is set using the height of the primary loop of the ‘seed’ character as a threshold.
4. Having found a suitable character the algorithm returns to step three using the recently found character as the new seed and simultaneously adding it to the logical line.
5. If no suitable character loop is found the logical line is sealed and the first character loop of the remaining list of loops is taken as the start seed loop of the next logical line.<sup>3</sup>

It was found that isolating the next logical level in the page analysis (logical words) was too difficult to achieve by the clustering technique. There are too many variables within the distribution of words in a line to effectively find a sound algorithm to isolate words simply using clustering. These variables include word gap distance, letter gap distance, letter width and which point of the word to take a reference from: start, middle or end. Even a value

---

<sup>2</sup>A ‘seed’ character can be defined as a character that has already been found.

<sup>3</sup>There is no guarantee that the list of logical lines found within a text block will be in produced reading order, but this is easily rectified by sorting the lines found based on their ‘y’ coordinates.

of  $k=1$  with all the above constraints plus a horizontal distance threshold based on the average word gap was not enough to achieve a high recognition success rate. Consequently, typographical algorithms were used to isolate words, after OCR was performed upon the text blocks. These typographical algorithms are summarised by Elliman[4].

## 4 Summary

We have produced good results on a variety of different page layouts including multi-column format. Our text decomposition technique is designed to be a preparation technique prior to processing the image with other specialised algorithms: optical character recognition; diagram recognition; table recognition; logical structure realisation. The algorithm implementation effectively filters out and partially classifies areas of the image.

Four figures are presented which show the system decomposing a page of text. Figure 1 displays the results of the vectorisation of the original TIFF image. In Figure 2 the vector loops have been grouped together into blocks of text. Very little processing remains in order to isolate logical paragraphs from this state. Figure 3 illustrates the lines found within the blocks of text. Figure 4 shows the inconsistent results of trying to find logical words using the  $k$ -clustering technique. The platform these results provide is a good start for the dictionary look-up algorithms that can be used to supplement word recognition.

### 4.1 Performance

There are two separate stages to the modified “ $k$ -clustering” algorithm: block isolation and decomposition of text blocks into lines and words. Processing of text blocks into lines is effective and efficient. However, the three neighbour clustering is computationally heavy. Furthermore, the worst case occurs when a full page image (for example a large photo) is passed to the three neighbour clustering algorithm for page decomposition. The number of neighbour comparisons is immense given that the vectorisation technique divides the page up into relatively small loops.

A decomposition technique based upon white space layout would nullify the worst case and improve the computational time[3]. However, the “ $k$ -clustering” technique is robust and produces sound results.

## References

- [1] L. O’Gorman. The document spectrum for page layout analysis. *IEEE Transactions on Pattern Recognition and Machine Intelligence*, 15(11):1162–1173, 1993.
- [2] H. S. Baird. The skew angle of printed documents. In *SPSE 40th Annual Conference and Symposium on Hybrid Imaging Systems*, pages 21–24, 1987.
- [3] T. Pavlidis and J. Zhou. Page segmentation and classification. *CVGIP: Graphic Models and Image Processing*, 54(6):484–496, 1992.
- [4] D.G. Elliman and I.T. Lancaster. A review of segmentation and contextual analysis techniques for text recognition. *Pattern Recognition*, 23(3):337–364, 1990.

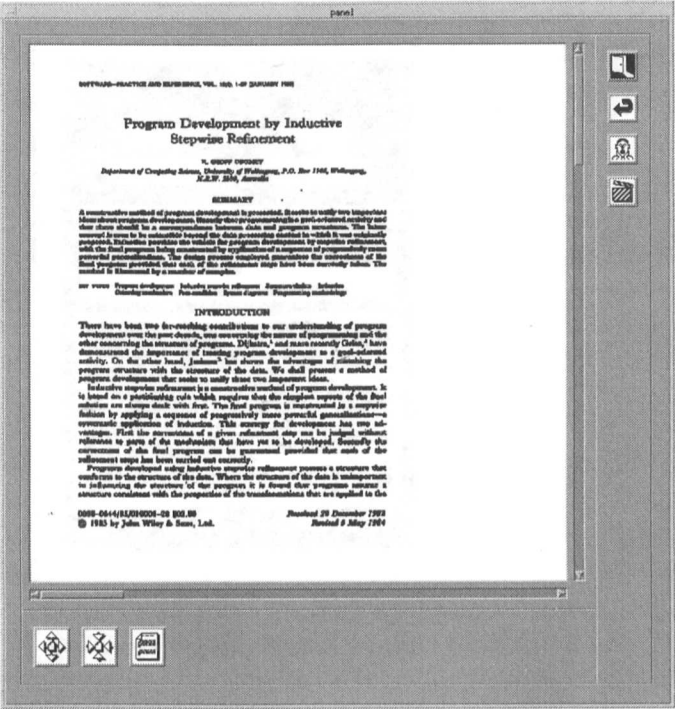


Figure 1: A normal vectorised document

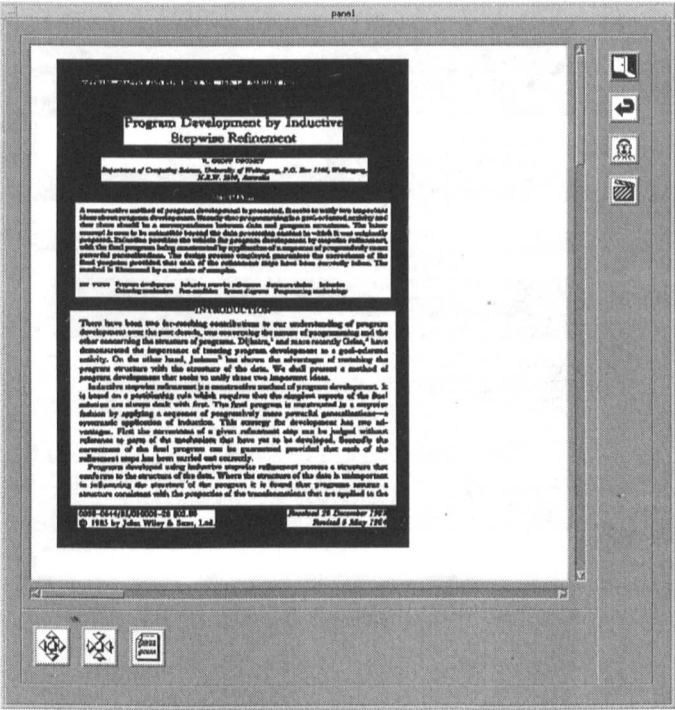


Figure 2: A vectored document with paragraphs isolated

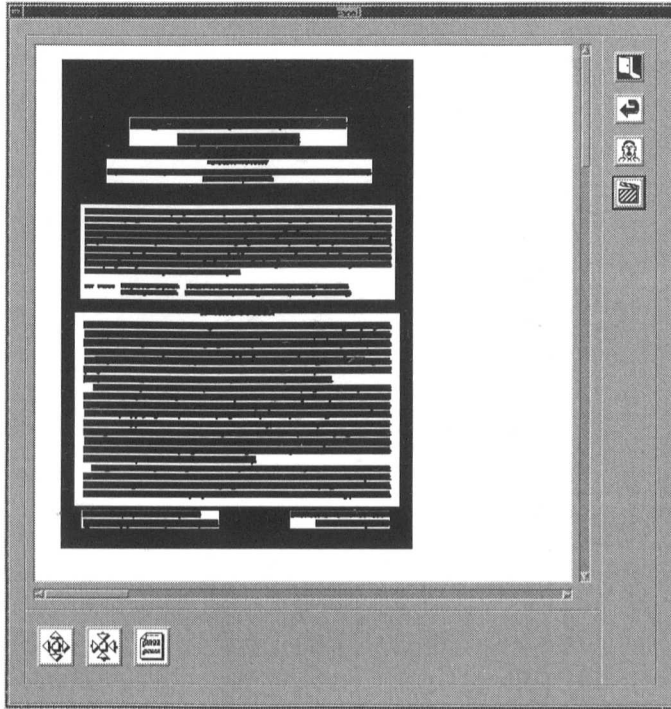


Figure 3: A vectored document with paragraphs and lines isolated

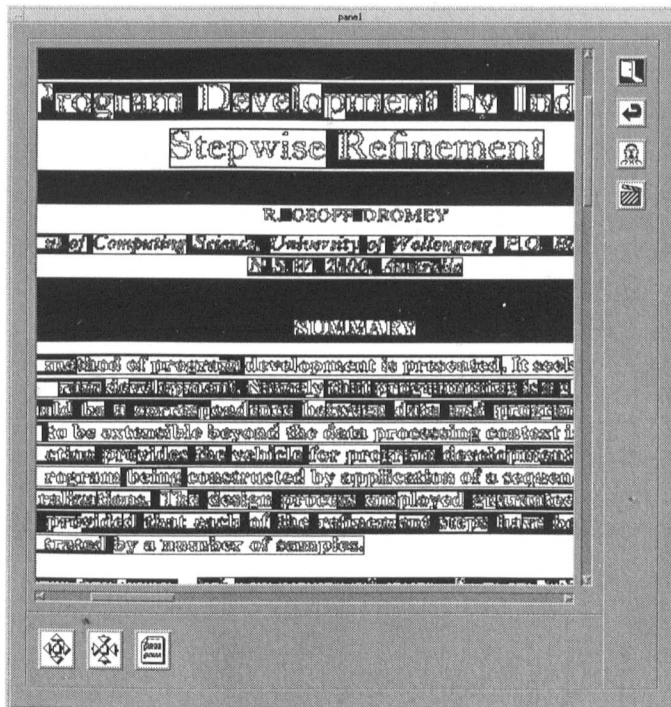


Figure 4: A vectored document with paragraphs, lines and words isolated